◆ 兼顾基础，重在提高与技巧

◆ 解惑答疑，提升系统开发水平

代码的力量

Windows CE 6 开发经典

邱小平　李　恬　雷群英　编著

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY
http://www.phei.com.cn

代码的力量

Windows CE 6 开发经典

本套丛书对编程语言、语法等基础知识讲得很少，重点在对语言的运用能力上。让读者看完书后，有一种“豁然开朗”的感觉，不只是简单地理解一种语言的语法，而是对一种技术的全面认识，并知道如何进行灵活运用。本书的目的主要是提升读者项目编程的知识，展现代码编写中的技巧与经验，让读者能更便捷地获取知识、积累经验，这也是本丛书的重要特色之一。

PHEI

天启星
http://www.tqxbook.com

策划编辑：祁玉芹
责任编辑：郭鹏飞
责任美编：王　嵩

本书贴有激光防伪标志，凡没有防伪标志者，属盗版图书。

ISBN 978-7-121-08573-4

9 787121 085734 >

定价：36.00元

代码的力量——

Windows CE 6 开发经典

邱小平　李　恬　雷群英　编著

電子工業出版社
Publishing House of Electronics Industry
北京·BEIJING

内 容 简 介

本书主要以 Windosw CE 6 为开发平台。用实例的方式介绍了 Eboot 的开发方法，给出了 FAT 文件系统，SD 卡驱动程序的详细源代码；以实例源码方式给出了流驱动的基本框架以及调用方法；以源码方式给出了串口通信 DLL 在 EVC 和.NET2005 中的实现方法；以源码方式给出了 WINCE 数据库的使用方法；以源码方式给出了 WINCE 打印机的使用方法。另外，本书给出了几个在实际 WINCE 开发工程应用当中可能会遇到的，较难的编程问题的解决方法：一是显示驱动；二是 WINCE6 上的关系数据库系统；三是 PCL 打印机。

本书不光基础知识完备，指导读者如何动手操作的内容也较多，并且覆盖面非常广，适合各个层次的读者。对于入门用户，可以依次阅读并动手实践。对于高级用户，可以考虑诸如 Eboot 高级开发，高性能显示绘图等相关内容。所以本书的读者对象包括：各个级别的、从事 WINCE 系统开发的人员，应用程序开发的人员，也可以作为大学相关专业高年级学生以及相关培训机构的教程。

未经许可，不得以任何方式复制或抄袭本书之部分或全部内容。
版权所有，侵权必究。

图书在版编目（CIP）数据

Windows CE 6 开发经典 / 邱小平等编著.—北京：电子工业出版社，2009.5
(代码的力量)
ISBN 978-7-121-08573-4

I. W… Ⅱ.邱… Ⅲ.窗口软件，Windows CE－程序设计 Ⅳ.TP316.7

中国版本图书馆 CIP 数据核字（2009）第 044872 号

策划编辑：祁玉芹
责任编辑：郭鹏飞
印　　刷：北京市天竺颖华印刷厂
装　　订：三河市鑫金马印装有限公司
出版发行：电子工业出版社
北京市海淀区万寿路 173 信箱　邮编 100036
开　　本：787×1092　1/16　印张：21.5　字数：550 千字
印　　次：2009 年 5 月第 1 次印刷
定　　价：36.00 元

凡所购买电子工业出版社图书有缺损问题，请向购买书店调换。若书店售缺，请与本社发行部联系，联系及邮购电话：（010）88254888。

质量投诉请发邮件至 zlts@phei.com.cn，盗版侵权举报请发邮件至 dbqq@phei.com.cn。

服务热线：（010）88258888。

前言

20 世纪 70 年代末，单片机开始应用于各个领域。经过数十年发展，以嵌入式智能设备为核心的后 PC 时代来临，嵌入式产品开始得以普及，嵌入式开发人才也成为当前最为紧缺的人才之一。

尽管微软的维纳斯计划过于超前而未获得太大的成功，但它造就的 Windows CE 操作系统，在十多年过后，已经在各种嵌入式操作系统中占据了极其重要的地位。由于 Windows CE 系统的稳定性、可靠性、实时性，还有媲美 Windows 操作系统的图形界面，成熟的开发平台、应用程序与桌面程序的一致性，使它获得众多开发者的青徕。Windows CE 系统开发门槛虽并不算高，但对入门用户还是有较高的障碍的，很多开发人员因此望而却步，不过本书正是为了解决这个问题而编写的。

本书共分 13 个章节，可以分成 3 个大的主题来阅读。

第 1 部分包括前 3 章：安装 Windows CE、定制 Windows CE 系统和运行 Windows CE 系统。这里介绍了如何获取全部的免费软件，如何自己制作 NK.bin 文件，如何在 VMware 虚拟机上运行 Windows CE 系统，以及在普通 PC 机上运行 WINCE 时的参考硬件配置与 BIOS 参数设置等。

第 2 部分包括第 4 到第 7 章，提供了与驱动相关的解决方法。第 4 章以实例的形式介绍了 Eboot 的开发过程，完整地实现了 FAT 文件系统与 SD 卡驱动，可以实现从 SD 卡上启动 WINCE 系统。第 5 章主要介绍了实现 Windows CE 文件系统的体系结构以及各组件的关系与实现方法。第 6 章主要介绍了 USB 驱动开发流程以及微软新提供的 USB 摄像头驱动，为在 WINCE 上实现摄像提供了基本方法。第 7 章主要介绍了 Windows CE 显示驱动的体系以及 Intel 针对 Windows CE 的显示驱动包，为 Windows CE 上高速绘图与 DirectX 的支持提供了基本解决方案。

第 3 部分包括第 8 章到第 13 章，主要涉及 Windows CE 应用程序开发方法。第 8 章简单介绍了 Windows CE 应用程序开发与 Windows 应用程序的一些异同之处。第 9 章简单介绍了

Platform Builder 开发驱动程序、应用程序的方法。第 10 章主要介绍了 EVC 开发串口通信 DLL 的方法。第 11 章主要介绍了在 WINCE 平台上用.NET（C#）精简版开发应用程序的方法，以实例方式介绍了.NET 精简版常用控件的使用方法，介绍了以托管方式开发串口通信 DLL 的方法。第 12 章介绍了在 WINCE6 平台上添加数据库支持的方法，以及数据库应用程序的开发方法。第 13 章介绍了在 WINCE6 平台上打印的解决方案，包括打印机驱动添加方法与打印应用程序开发方法。

本书实用性强，并且覆盖面非常广，适合各个层次的读者。对于入门用户，可以依次动手实践。对于高级用户，可以考虑诸如 Eboot 高级开发，高性能显示绘图等相关内容。本书尽量避免纠缠于操作系统内核运行机制、Windows CE 系统源代码等相关问题上，重点放在一些可以上手实践操作的实例，以及某些大型问题的解决方案上面。

对于技术人员来说，很多问题只需要一句话提示即可。本书所说的几个核心问题实际上可以归纳为以下几句话：X86 平台上 Windows CE 的高性能绘图与显示，Intel 提供了驱动；X86 平台上的摄像头驱动微软已经提供，Windows CE 6 数据库可以手工移植；Windows CE 6 上也可以直接驱动所有 PCL 系列打印机。

本书写作过程当中，要特别感谢两位好友，蒋辉和唐英勇。两位均有十多年的软硬件项目开发经验，对 WINCE 的研究非常深入。本书的大部分实例源码均由两人提供并测试。作为编者，真正付出的时间精力还没有这两位朋友多，在此向两位的付出深表感谢。读者朋友如有什么任何疑问和建议，可以到如下网站的本书讨论区同笔者一起探讨：

http://www.pubeta.com

如果您在学习本书的过程中遇到问题，请与我们联系。

我们的 E-mail 地址为 guopengfei @phei.com.cn。

编著者

2009 年 2 月

目录

CONTENTS

第 1 章 安装 Windows CE 6 1

1.1 获得免费软件 1
1.2 安装 VS2005 1
1.3 安装 MSDN 文档 3
1.4 安装 Platform Builder 6.0 5
1.5 安装 VMware 虚拟机软件 8

第 2 章 定制 Windows CE 系统 9

2.1 定制 X86 平台的最小图形系统 9
2.2 定制 X86 平台的完整系统 14

第 3 章 运行 Windows CE 系统 25

3.1 配置 VMWare 运行环境 25
3.2 配置 X86 硬件运行环境 39
3.3 运行时调试操作系统 47

第 4 章 BootLoader 的开发 51

4.1 BootLoader 介绍 51
4.2 EBoot 开发例程 53
4.3 其他硬件平台的 BootLoader 149
4.3.1 vivi 介绍 149
4.3.2 U-Boot 介绍 150
4.4 BIOS BootLoader 配置方法 151

第 5 章 文件系统 153

5.1 Windows CE 文件系统体系结构 153
5.2 实现 Windows CE 6 的硬盘文件系统 157
5.3 实现 Windows CE 6 的 USB 文件系统 160

5.4 实现 Windows CE 6 的光盘文件系统 161

第 6 章 USB 驱动 163

6.1 USB 驱动开发介绍 163
6.2 USB 摄像头驱动介绍 175
6.2.1 USB Video Class 基础概念 175
6.2.2 USB Video Class 协议结构 176
6.3 微软的 USB Video Class 驱动使用 179

第 7 章 显卡驱动 189

7.1 WINCE 显卡驱动编程介绍 189
7.2 Intel 的 IEGD 显卡驱动包 195

第 8 章 WINCE 程序开发入门 225

8.1 WINCE 下 Win32 应用程序介绍 225
8.2 WINCE 下 32 位 API 的调用 226

第 9 章 用 Platform Builder 开发程序 231

9.1 Platform Builder 开发 DLL 的流程 231
9.2 用 Platform Builder 开发标准流驱动框架 232
9.3 Platform Builder 调用流驱动程序示例 241

第 10 章 EVC 程序开发 259

10.1 EVC 应用程序开发入门 259
10.2 EVC 开发串口通信 DLL 266

第 11 章 .NET 应用程序开发 283

11.1 .NET2.0 精简版基本控件使用方法 283
11.2 C#开发串口通信例程 289

第 12 章 WINCE 6 数据库开发 309

12.1 Windows CE 6 的数据库版本 309
12.2 .NET 操作 SqlServerCE 的数据库类 313
12.3 操作 SqlServerCE 数据库实例 322

第 13 章 在 Windows CE 上进行打印 325

13.1 PCL 打印驱动的介绍 325
13.2 EVC 打印程序实例 327

第 1 章　安装 Windows CE 6

众所周知，Windows CE（本书以下简称 WINCE）是一个独立的操作系统，它的开发过程实际上是一个交叉编译过程，更类似于单片机、DSP、ARM 等的开发。WINCE 的开发并不是安装一个 WINCE 系统，然后再在此系统上安装相应的开发工具来开发应用程序。实际上，WINCE 操作系统是“编译”出来的，而不是“安装”出来的，没有从事过 WINCE 开发或者相关嵌入式开发的读者，这就是需要理解的第一个问题。

1.1　获得免费软件

从事嵌入式相关开发，必要的软硬件配置也是必不可少的。有条件拥有 WINCE 开发平台当然最好，但并不是每一个人都有足够的资金、条件配备完备的软硬件开发环境。那么，另一个办法就是在家用 PC 平台上，利用各种免费的软件来搭建一个学习 WINCE 操作系统与应用程序开发的虚拟环境。

软件可以使用微软公司提供的 120 天试用版软件，包括.NET 2005 和 WINCE6，它们都可以在微软公司的网站上下载。读者也可以订购 DVD——这个所花的费用相信是大家能够接受的。目标硬件平台的解决也很简单，很多“古老”的 X86 PC 机都可以完全兼容 WINCE 平台。不能正常兼容 WINCE 的，可以采用 VMWARE 或者是 VIRTUAL PC 虚拟机来作为执行 WINCE 系统的目标机器。

建议：各种软件全部安装在 C 盘，以便于系统备份与还原；分区时给 C 盘预留 30 GB 以上的空间。

读者在 PC 机上搭建 WINCE 系统的学习平台，可以极大地降低 WINCE 的学习门槛，采用虚拟平台就可以完成所有的实验过程。

WINCE4.2、WINCE5.0 等版本在开发操作系统时，都有一个独立的 Platform Builder 平台。WINCE6 的安装同以前版本的 WINCE 不同，Platform Builder 不再是单独的应用平台，而是作为.NET 2005 的一个插件存在。所以，必须首先安装.NET 2005，然后才能安装 WINCE。

微软下载中心中，.NET 2005 及相关软件、补丁的下载地址为：

http://www.microsoft.com/downloads/search.aspx?displaylang=zh-cn

读者可以在如下网站下载一套完整的、180 天免费试用版的 Windows Embedded CE 6.0：

http://www.microsoft.com/windows/embedded/eval/wince/default.mspx

读者注意，安装时需要在 http://www.microsoft.com/windows/embedded/eval/trial.mspx 注册取得序列号。

1.2　安装 VS2005

WINCE6 的操作系统编译平台是作为 Visual Studio 2005（本书以下简称 VS2005）的一个

插件存在的，首先必须安装 VS2005 才能正常使用。完整安装 VS2005 程序需要 4 GB 左右的空间。

步骤 1 放入安装光盘并运行安装程序，在安装对话框中单击“安装 Visual Studio 2005”链接，如图 1-1 所示。

图 1-1

步骤 2 选择“完全”安装产品所有功能，建议不修改安装的默认路径，直接安装在 C 盘下面，如图 1-2 所示。

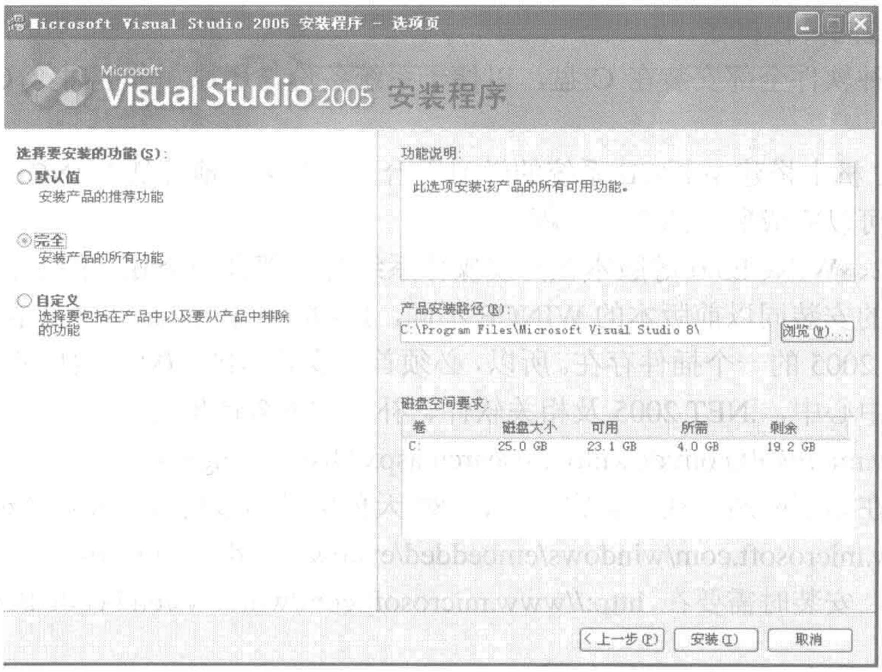


图 1-2

步骤 3 安装完成后，有可能出现 Office 依赖项警告。如果不需要开发 Office 相关程序，可以直接单击“完成”按钮，不作处理，如图 1-3 所示。如果有必要的话，就需要根据提示安装 Office 补丁。

图 1-3

1.3　安装 MSDN 文档

步骤 1 VS2005 安装完成后，会弹出窗口提示安装 MSDN，如图 1-4 所示。MSDN 是最好的学习资料，建议完全安装，这需要 2 GB 左右的硬盘空间。

图 1-4

步骤 2 选择接受协议条款，然后单击“下一步”按钮，到下一个安装界面，如图 1-5 所示。

步骤 3 根据需要输入用户名和单位名称，也可以随意填写，完成后单击“下一步”按钮，如图 1-6 所示。

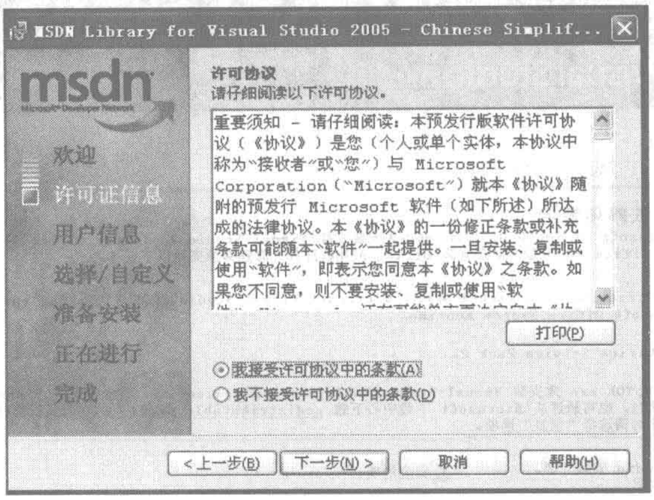


图 1-5

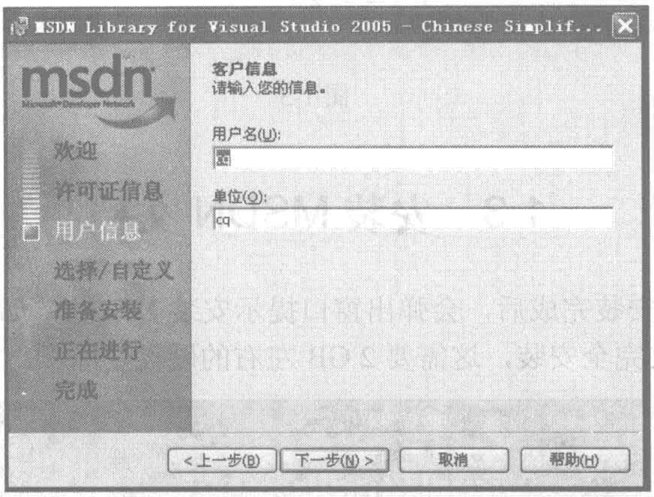


图 1-6

步骤 4 选择“完全（建议）”安装类型会将帮助文档全部安装到 C 盘，以便于备份，完成后单击“下一步”按钮，如图 1-7 所示。

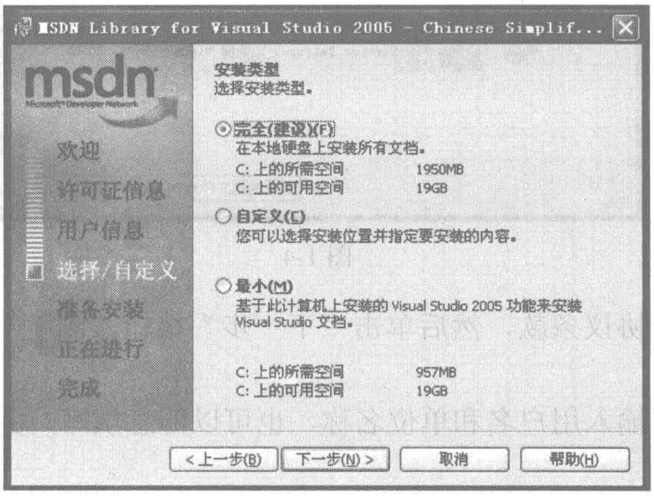


图 1-7

步骤 5 文件复制完毕，整个安装过程就结束了，单击“完成”按钮即可，如图 1-8 所示。

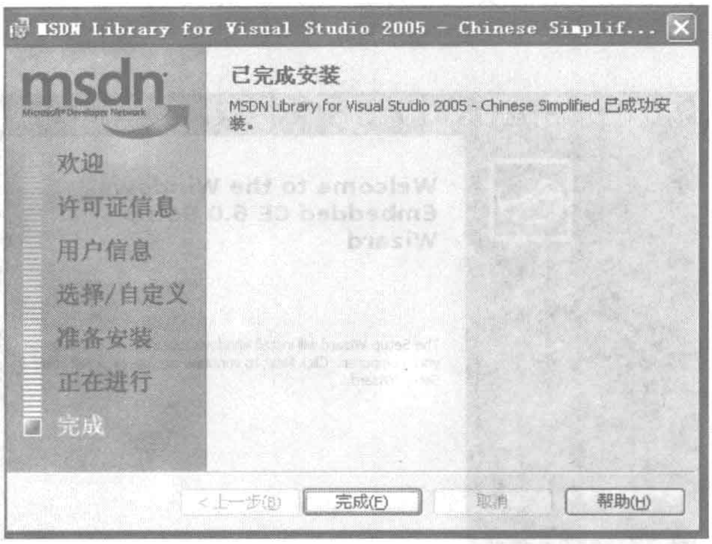


图 1-8

VS2005 与 MSDN 安装完成之后，启动界面如图 1-9 所示。此时，读者可以看到，这里还没有出现 WINCE 的相关选项。读者可以按照自己的习惯设置开发环境的界面布局，也可以自行调整。

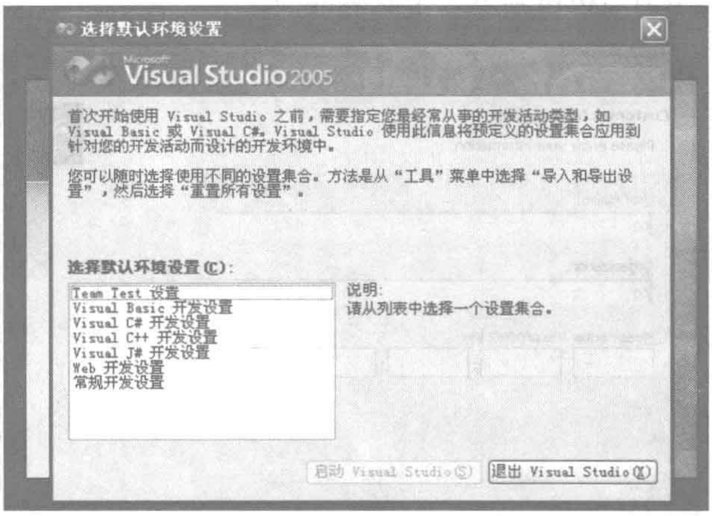


图 1-9

1.4　安装 Platform Builder 6.0

Platform Builder 6.0 是 VS2005 的一个插件，产品序列号可以在前文提供的微软网站上进行申请。为了完成本书提供的各项实验，安装时至少需要选择 X86 的完整 BSP 以及所有源码。它需要 6 GB 左右的硬盘空间，建议不要另外选择安装路径，全部安装至 C 盘即可。WINCE 是一个非常稳定的操作系统，但是 Platform Builder 却是一个异常脆弱的平台，稍有不慎就可能破坏整个平台。而且，一旦平台被破坏，只有重装操作系统、重装 VS2005、重装 Platform

Builder 6.0 才能让平台重新正常工作。笔者也是在经历数次惨痛教训之后，才发现其中的问题。最好的解决之道就是对整个系统进行备份（C 盘），一旦被破坏只需要恢复系统即可。

步骤 1 放入安装光盘，单击安装程序，在弹出的安装界面上单击“Next”按钮，如图 1-10 所示。

图 1-10

步骤 2 在接下来的安装界面需要用户输入序列号，读者填入申请到的序列号即可，如图 1-11 所示，单击“Next”按钮到下一个安装界面。

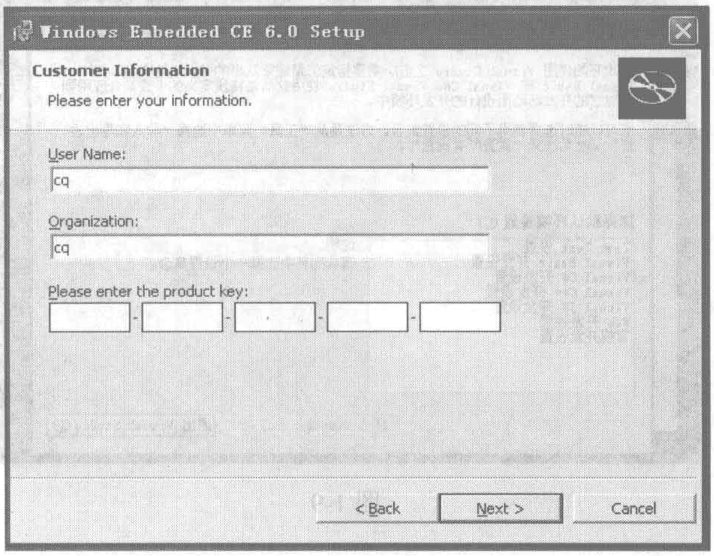


图 1-11

步骤 3 选择接受许可协议，然后单击“Next”按钮，如图 1-12 所示。

步骤 4 至少选择如图 1-13 所示的组件，如果需要开发其他硬件平台的 WINCE 系统，还需要增加选择对应的硬件平台，如图 1-13 所示。完成后，单击“Next”按钮到下一个安装界面。

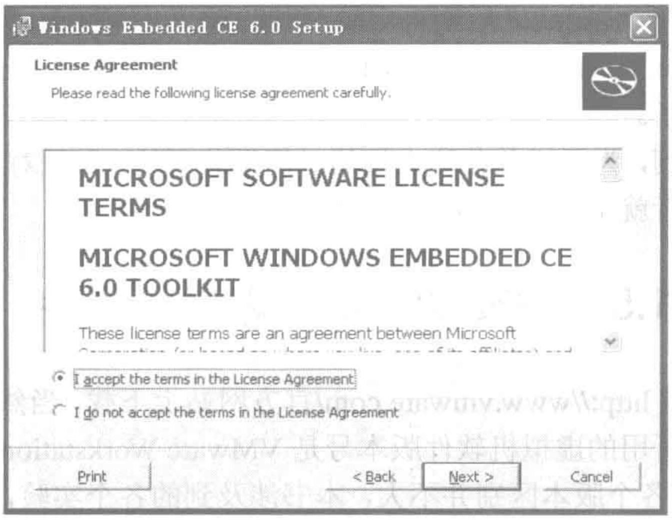


图 1-12

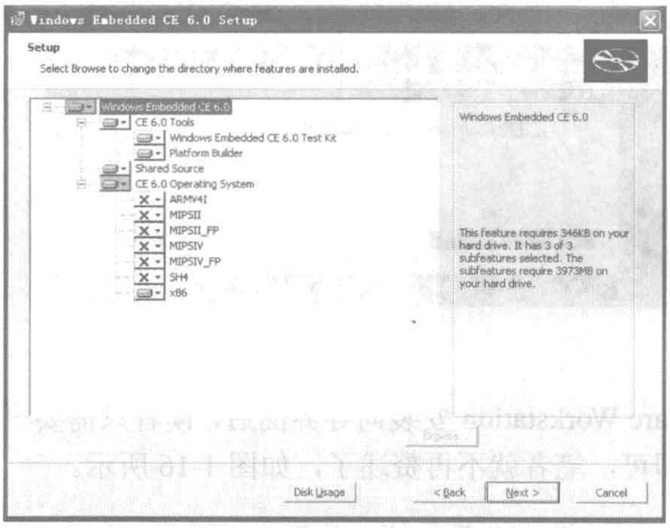


图 1-13

步骤 5 耐心等待安装程序复制文件，直至安装结束，根据硬件配置的不同，这一过程可能花费数分钟甚至数小时，如图 1-14 所示。

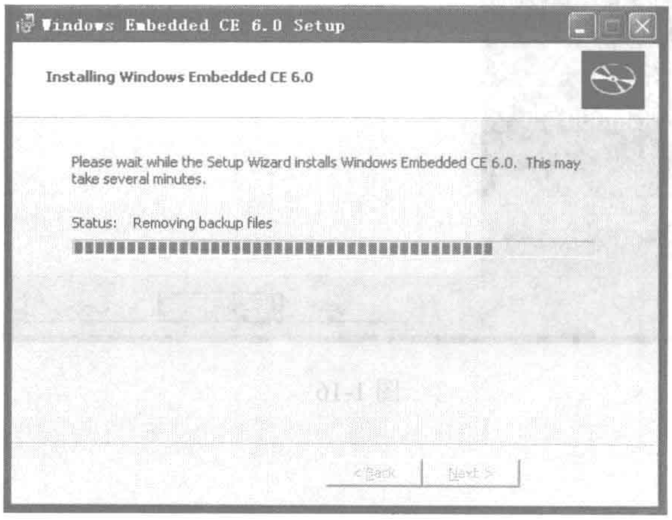


图 1-14

Platform Builder 6.0 安装完成之后，再启动 VS2005，在图 1-9 所示的默认环境设置当中就多了一个“Platform Builder Development Setting”选项，这说明 Platform Builder 6.0 已经正常安装，可以开始使用了。

笔者在前面已经提到，为了避免花费大量的时间重装系统，必须对系统进行备份。所以，在这里安装完成后，读者就可以使用 Ghost 软件对 C 盘进行备份了。

1.5 安装 VMware 虚拟机软件

VMware 软件可以在 http://www.vmware.com/官方网站上下载，当然也可以在其他安全的下载站点上下载。笔者所用的虚拟机软件版本号是 VMware Workstation v6.5 Build 110068 Beta。实际上，VMware 各个版本区别并不大，本书涉及到的各个实验，VMware 各个版本几乎可以通用。VMware 的使用方法将在后面详细介绍，这里先介绍安装过程。

步骤 1 双击安装程序，会出现如图 1-15 所示的安装准备进度条。

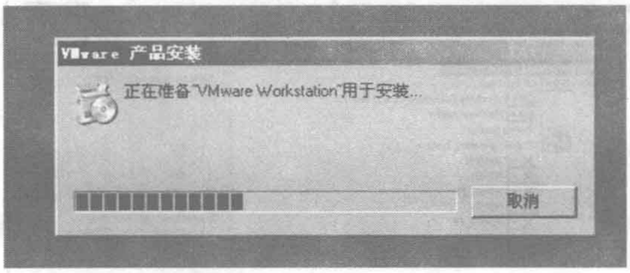


图 1-15

步骤 2 进入 VMware Workstation 安装向导界面后，读者只需要一路单击“Next”按钮，使用默认设置完成安装即可，笔者就不再赘述了，如图 1-16 所示。

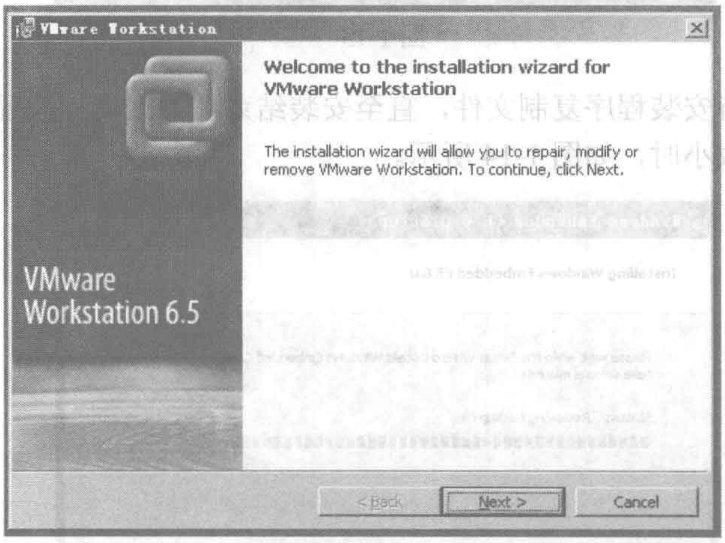


图 1-16

第 2 章　定制 Windows CE 系统

定制 WINCE 操作系统，几乎是每一个 WINCE 开发人员的必修课。此事说来极易，上手也非常简单，但是此项工作易懂难精，涉及到大大小小的数十个问题。并且定制系统是联结硬件与应用程序的纽带。可以在操作系统定制过程中辅助查找硬件的问题，也可以辅助查找应用程序的问题。通常，整个嵌入式平台不能正常运行，需要定位故障点时，可以先在成熟平台上定制一个确保完全正确的 WINCE 操作系统，保证中间层运行稳定之后再往底层查找硬件的问题，往上层查找应用程序的问题。根据 BSP 的不同，WINCE 系统定制过程稍有差异，但相差并不太大。本章以 WINCE 6 平台自带的 BSP 为例进行说明。

2.1　定制 X86 平台的最小图形系统

定制 X86 最小图形系统，其目的有两个：一是检验开发平台是否正确，即操作系统、VS2005 和 Platform Builder 平台是否被破坏；二是确保没有其他组件依赖关系“干扰”的情况下，以最简单的方式编译一个“干净”的系统。

步骤 1 打开 VS2005，选择“文件”→“新建”→“项目”选项，打开“新建项目”对话框，在“项目类型”列表框中选择“Platform Builder for CE 6.0”选项，新建项目名称为“OSDesign1”，项目保存的路径为默认路径，然后单击“确定”按钮，如图 2-1 所示。请读者注意，这个路径在项目生成之后是不可更改的，如果再次编译项目时，路径发生变化将会出错。

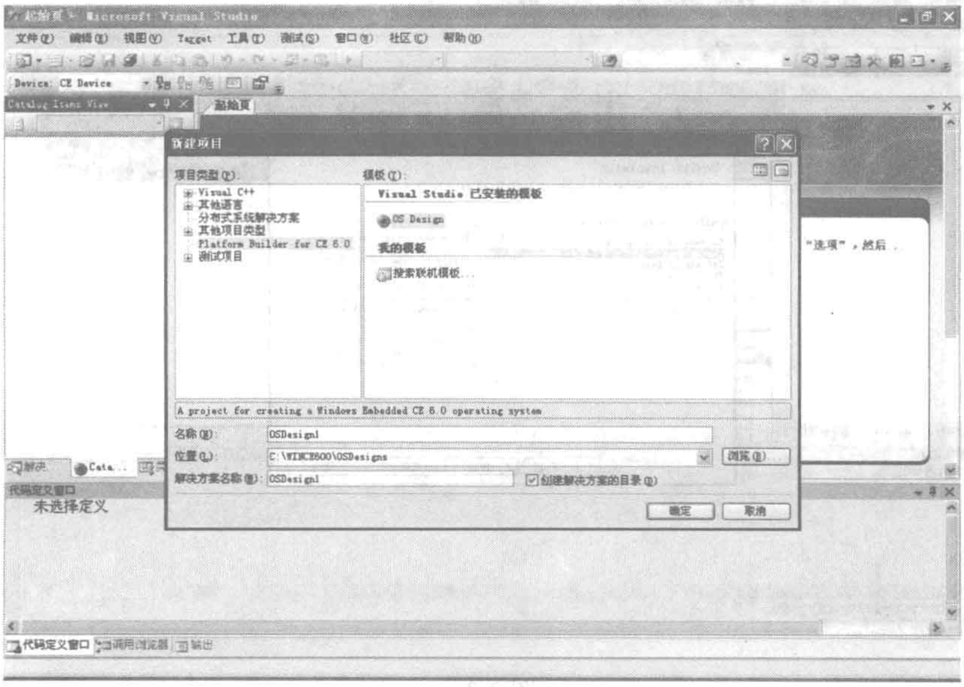


图 2-1

步骤 2 在弹出的“Windows Embedded CE 6.0 OS Design Wizard”向导界面中单击“下一步”按钮，进入 BSP 板级支持包选择对话框，这里选中参考平台“CEPC：x86”，单击“下一步”按钮，如图 2-2 所示。

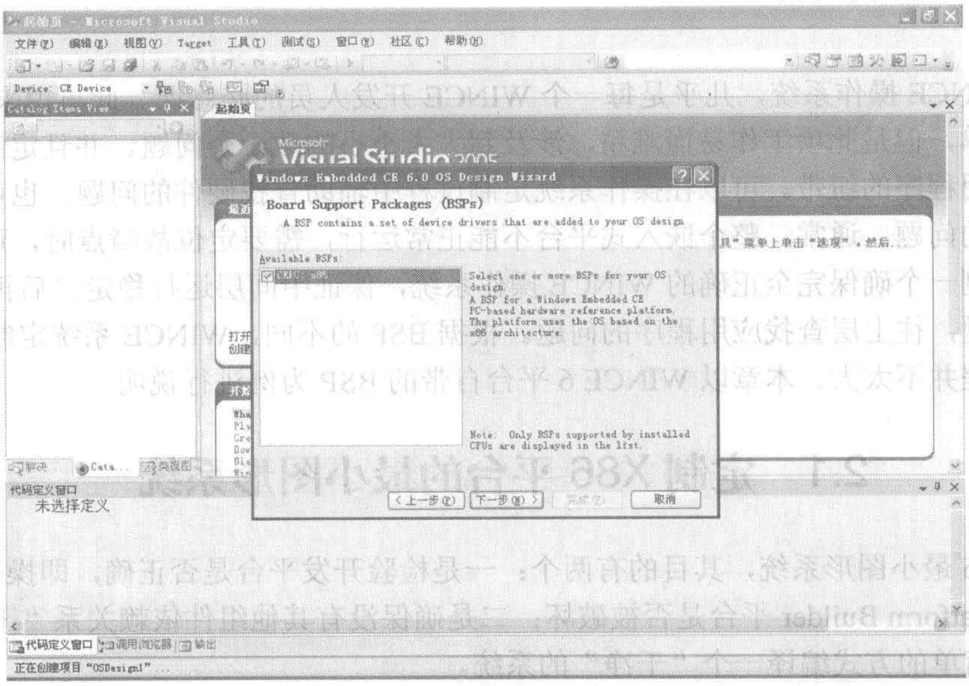


图 2-2

步骤 3 在接下来的平台配置选择界面中，选择“Custom Device”选项，全部自定义选择组件，单击“下一步”按钮，如图 2-3 所示。

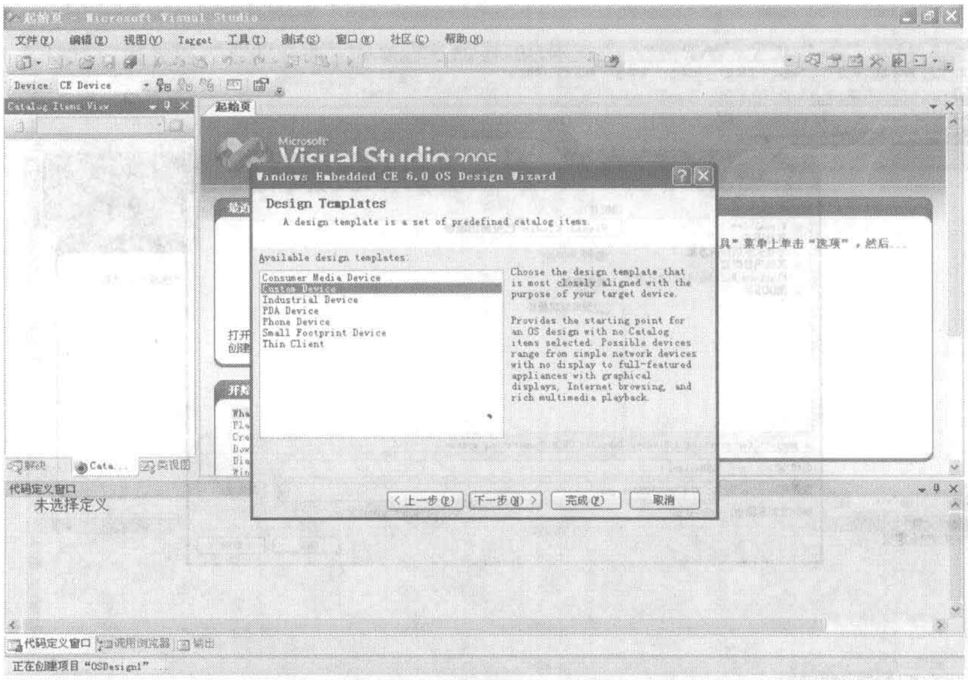


图 2-3

步骤 4 接下来的向导步骤，读者可以保持系统默认，一路单击“下一步”按钮直到图

2-4 所示的界面。为了定制最小图形系统，这里只需要选择“Strand Shell”一项即可。根据 PLATFORM BUILDER 平台的“组件自动侦测”功能，编译系统将会自动将其他所需要的组件关联编译。

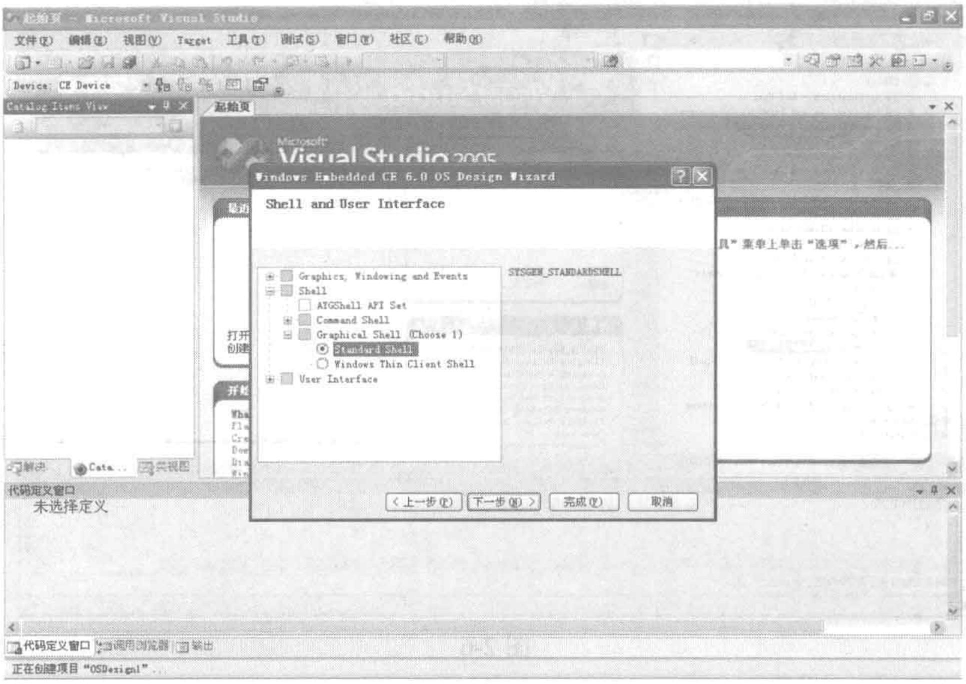


图 2-4

步骤 5 继续保持系统默认，单击“下一步”按钮完成所有组件选择，最后单击“完成”按钮，如图 2-5 所示。

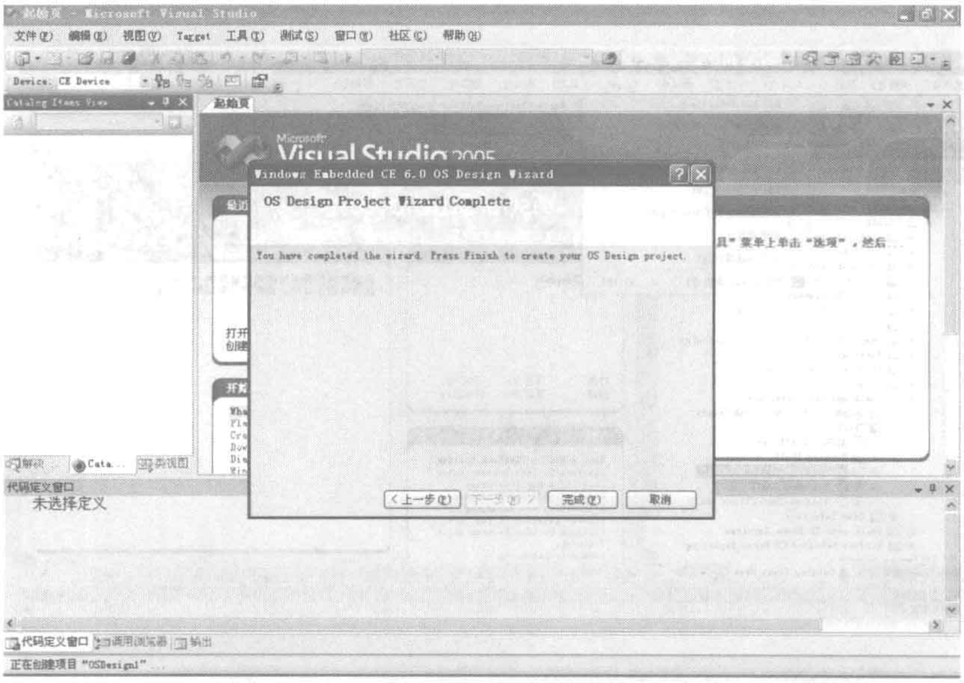


图 2-5

步骤 6 如图 2-6 所示，将编译版本设置为“Release”版本。如果不需要联机调试，最

好每次都编译为 Release 版本。因为 Debug 版本会输出很多调试信息，运行速度较慢，目标文件也要庞大得多，会影响最终运行的效果。

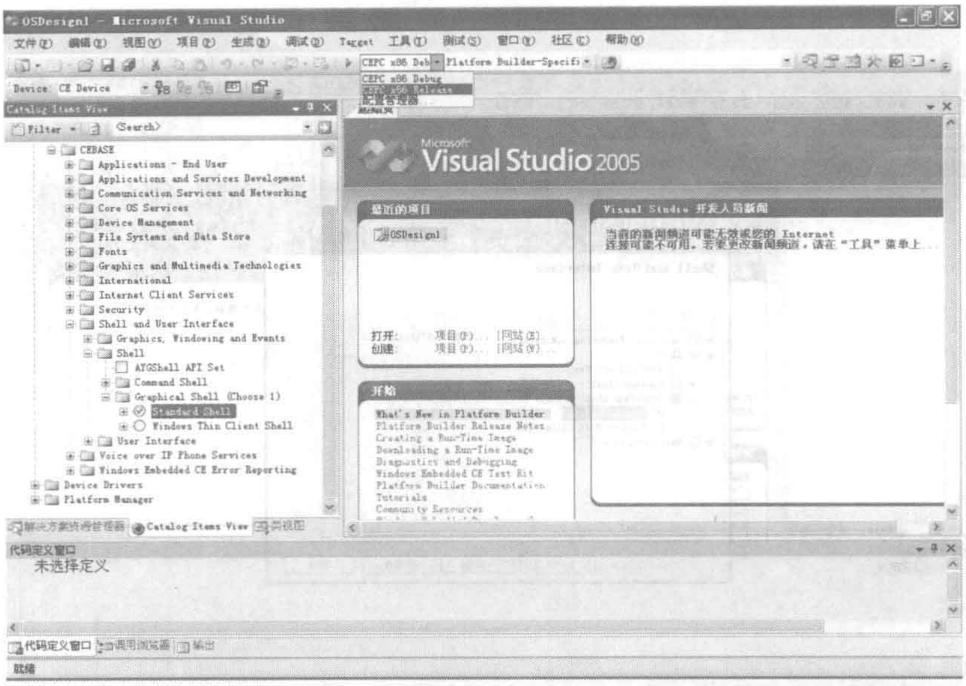


图 2-6

步骤 7 配置其他编译属性。如图 2-7 所示，在 VS2005 中，选择“项目”→“OSDesign1 属性”选项，弹出窗口如图 2-8 所示的对话框，选择“Build Options”选项，各项按需选择。

步骤 8 若需要添加环境变量，选择“Envirenment”选项，弹出如图 2-9 所示的对话框，可在其中添加环境变量（最小图形系统的配置过程不需要添加新的环境变量，后文将用到，不再重复）。

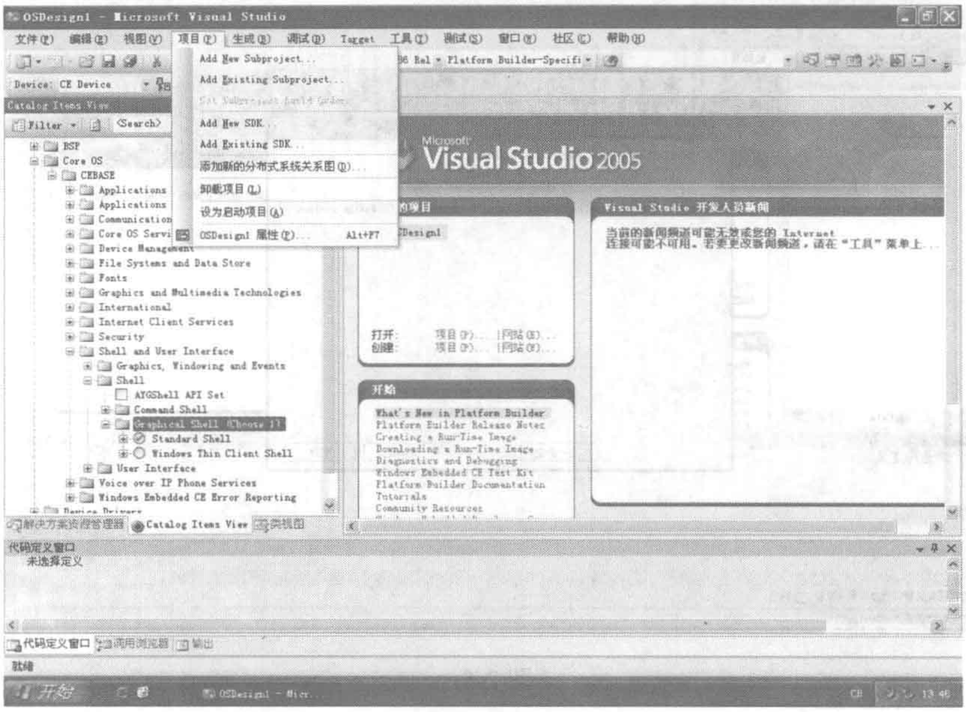


图 2-7

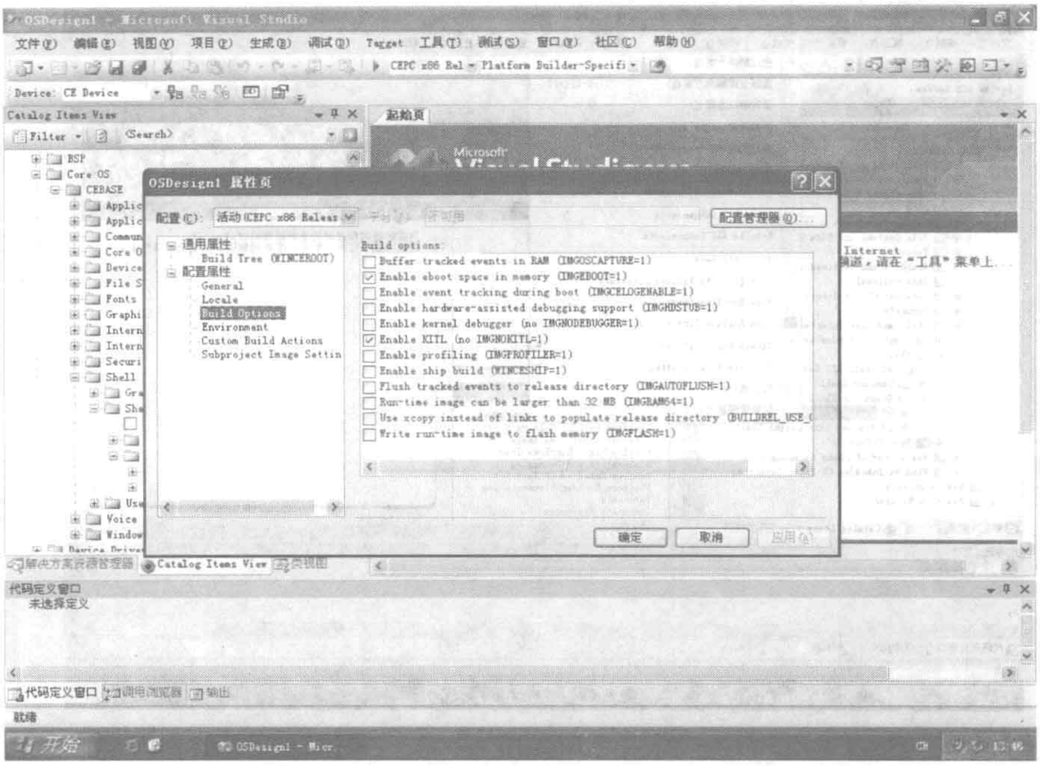


图 2-8

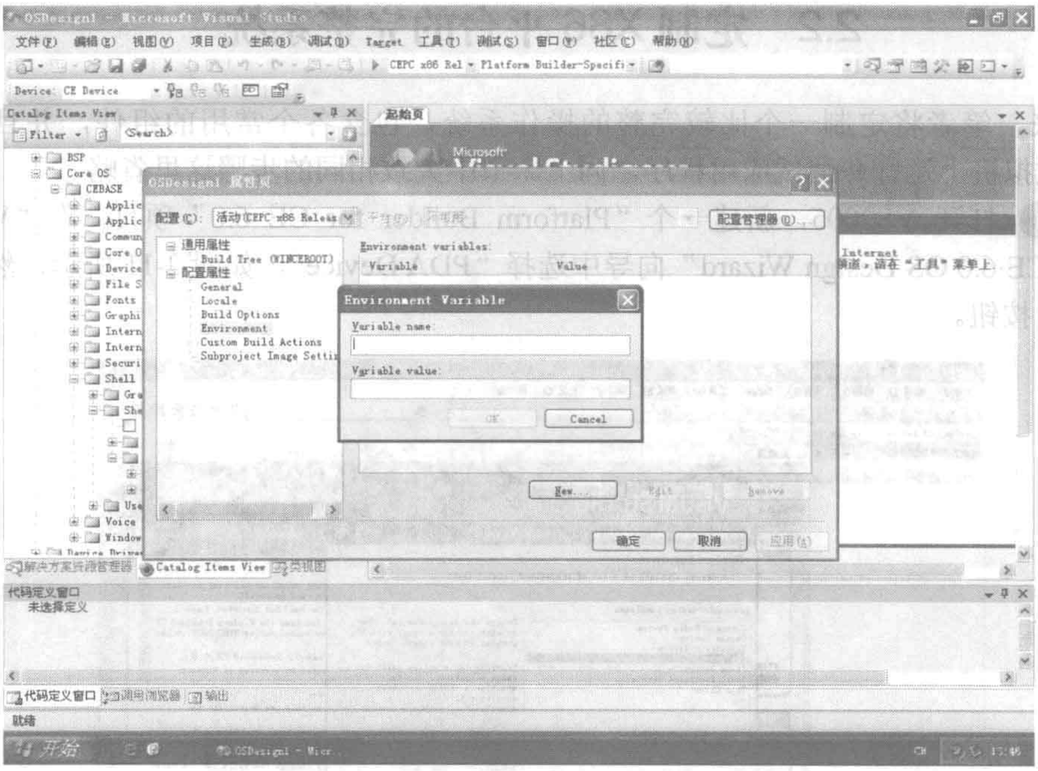


图 2-9

步骤 9 打开 VS2005，选择“生成”→“生成解决方案”或者“生成 OSDesign1”选项，编译 WINCE6 操作系统工程，如图 2-10 所示。根据硬件配置不同，这一过程可能花费数十分钟甚至数小时。

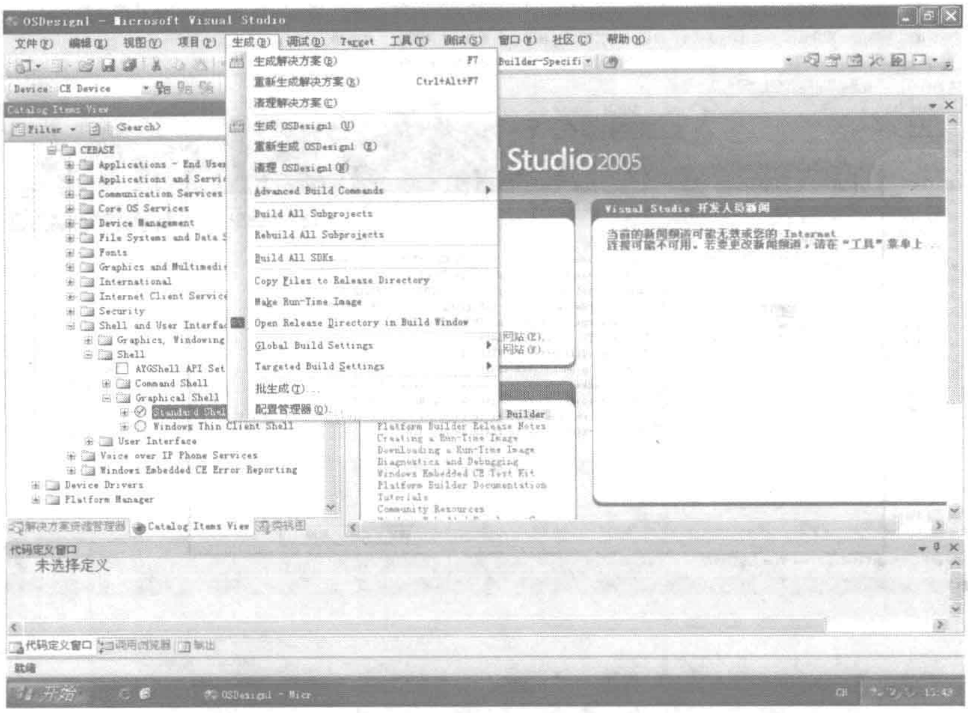


图 2-10

2.2 定制 X86 平台的完整系统

接下来，笔者将定制一个比较完整的操作系统，包含各个常用的组件，并且可以在 VMWare 虚拟机上运行和调试应用程序。同上一节中大致相同的步骤这里省略。

步骤 1 打开 VS2005，新建一个“Platform Builder for CE 6.0”项目，在“Windows Embedded CE 6.0 OS Design Wizard”向导中选择“PDA Device”，如图 2-11 所示，然后单击“下一步”按钮。

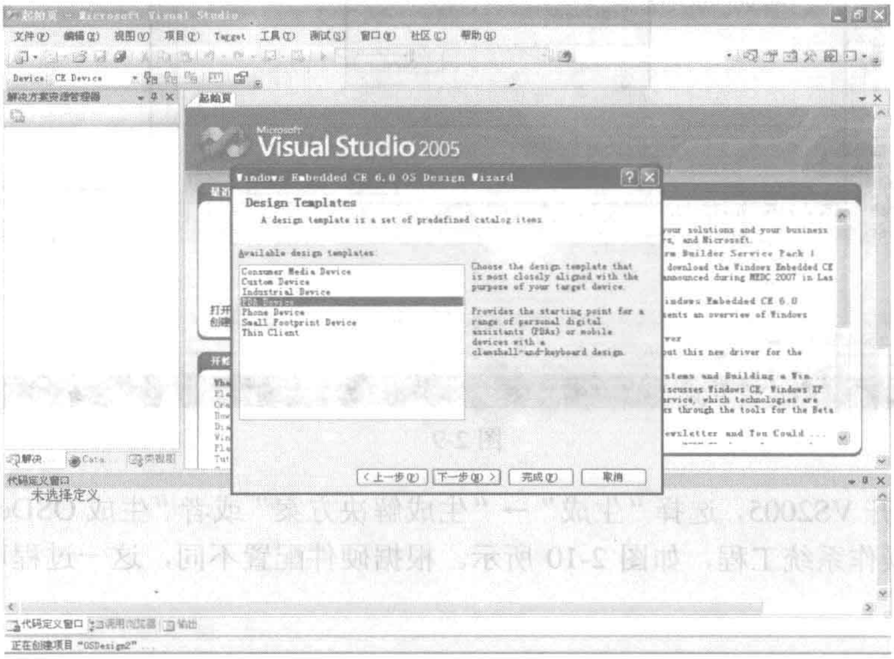


图 2-11

步骤 2 如图 2-12 所示，在本向导页中选择“Enterprise Web Pad”选项，然后单击“下一步”按钮。

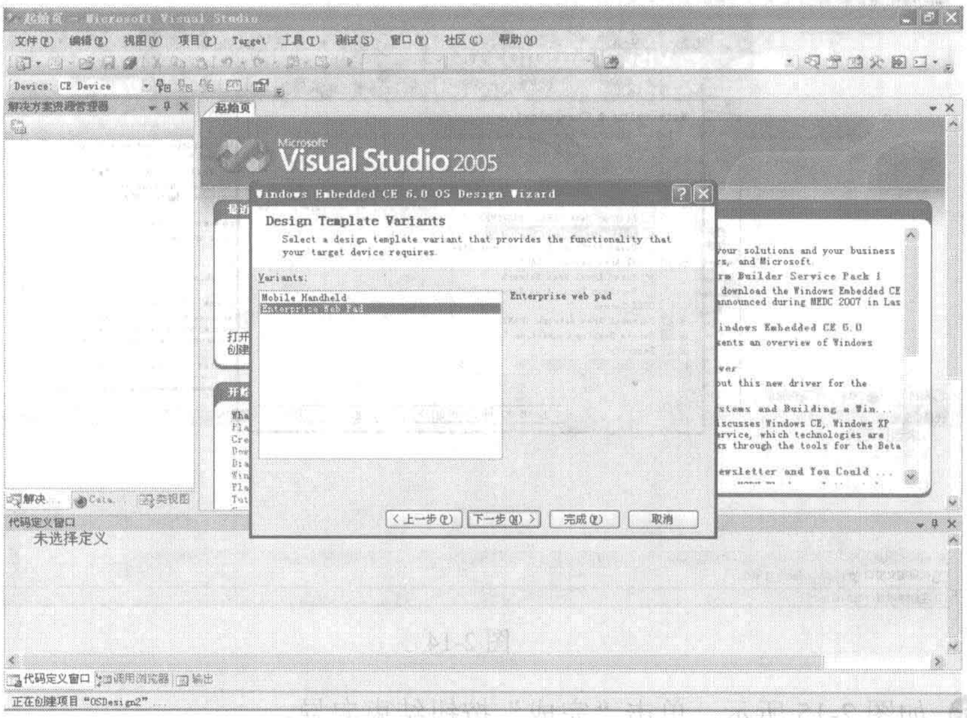


图 2-12

步骤 3 如图 2-13 所示，按照用户所需要功能选择必要的组件，某些情况下建议将 DCOM 选项选中，然后单击“下一步”按钮。

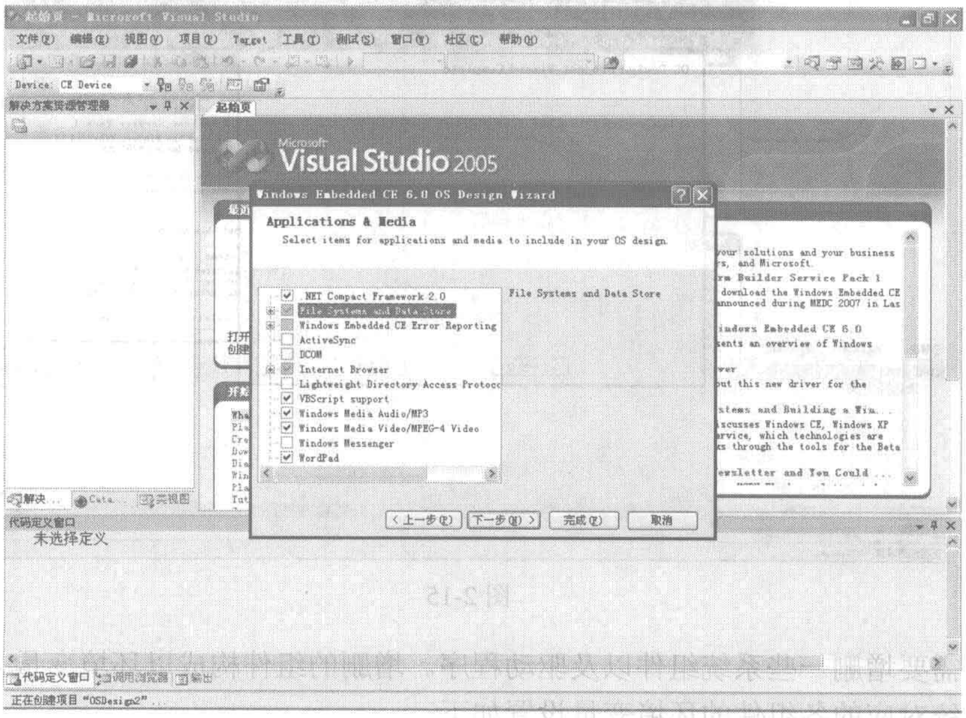


图 2-13

步骤 4 如图 2-14 所示，选择所需要的网络功能，建议将“TCP/IPv6 Support”和“Wired Local Area NetWork”选项勾选，然后单击“下一步”按钮。

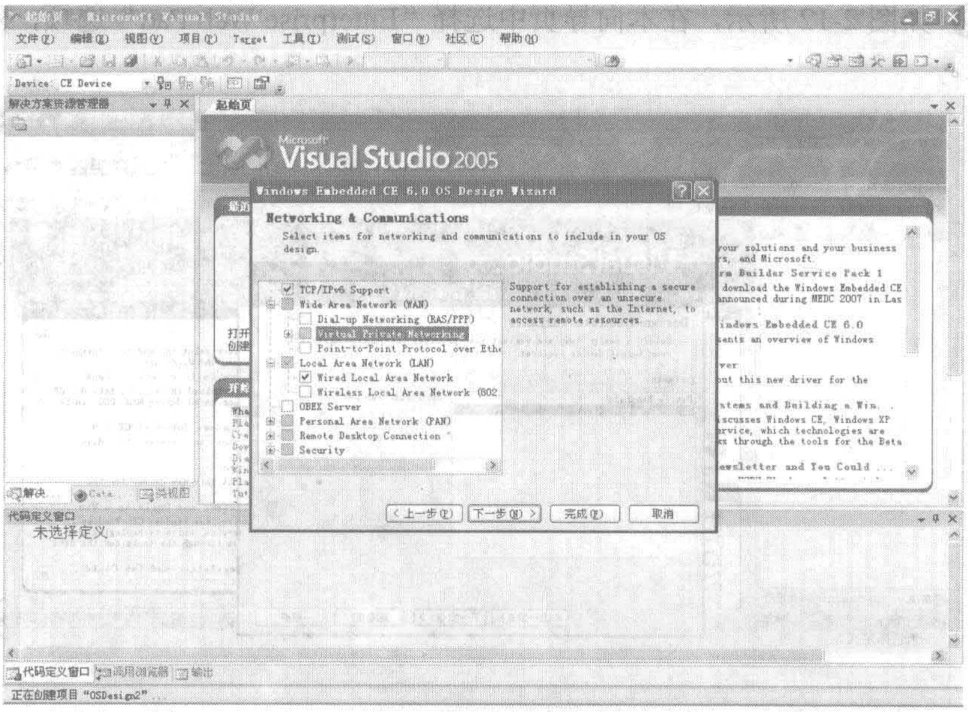


图 2-14

步骤 5 如图 2-15 所示，单击“完成”按钮结束向导。

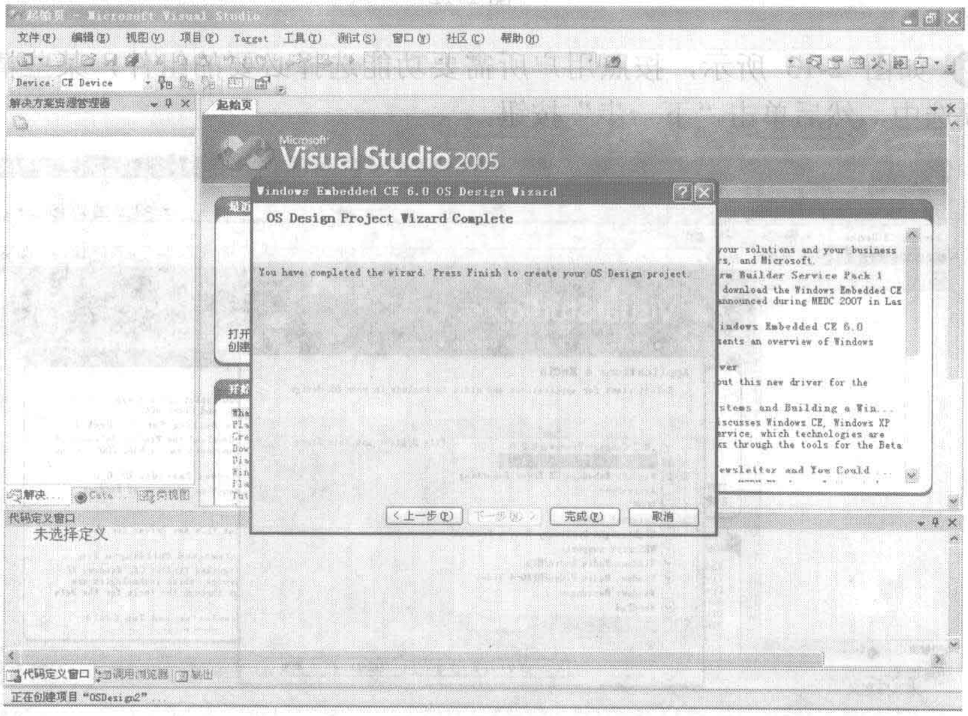


图 2-15

然后，需要增删一些系统组件以及驱动程序。增删的组件构成以环境变量方式给出。操作系统对应的各组件的环境变量设置如下。

```
@echo off
REM Initial environment configuration
```

```
set _PB_INSTALL_ROOT=C:\PROGRA~1\MI0D56~1\6.00
set USING_PB_WORKSPACE_ENVIRONMENT=1
set _WINCEROOT=C:\WINCE600
set
_FLATRELEASEDIR=C:\WINCE600\OSDesigns\OSDesign2\OSDesign2\RelDir\CEPC_x86_Rele
ase
set LOCALE=0804
set
_PROJECTROOT=C:\WINCE600\OSDesigns\OSDesign2\OSDesign2\Wince600\CEPC_x86

REM Workspace and configuration variables

set PBWORKSPACE=C:\WINCE600\OSDesigns\OSDesign2\OSDesign2\OSDesign2.pbxml
set PBWORKSPACEROOT=C:\WINCE600\OSDesigns\OSDesign2\OSDesign2
set PBCONFIG=CEPC x86 Release

REM Call wince.bat

call C:\WINCE600\public\COMMON\OAK\MISC\wince.bat x86 OSDesign2 CEPC

REM Make sure all build options are turned off

set IMGNODEBUGGER=
set IMGNOKITL=
set IMGEBOOT=
set IMGFLASH=
set IMGPROFILER=
set IMGCELOGENABLE=
set IMGAUTOFLUSH=
set IMGOSCAPTURE=
set BUILDREL_USE_COPY=
set WINCESHIP=
set IMGRAM64=
set IMGHDSTUB=

REM Anchored features

set SYSGEN_AUDIO=1
set SYSGEN_AUTH=1
set SYSGEN_AUTH_NTLM=1
set SYSGEN_AUTH_SCHANNEL=1
set SYSGEN_CERTS=1
set SYSGEN_CMD=1
set SYSGEN_CONNMC=1
set SYSGEN_CONSOLE=1
set SYSGEN_CRYPTO=1
set SYSGEN_CURSOR=1
set SYSGEN_HELP=1
set SYSGEN_IABASE=1
set SYSGEN_IE_DXTRANS=1
set SYSGEN_IE_IMGUTIL=1
set SYSGEN_IE_JSCRIPT=1
```

```
set SYSGEN_IE_P3P=1
set SYSGEN_IE_PNGFILT=1
set SYSGEN_INETCPL=1
set SYSGEN_MLANG=1
set SYSGEN_MLANGALL=1
set SYSGEN_MSHTML=1
set SYSGEN_MSHTML_HTMLAPP=1
set SYSGEN_MSXML_DOM=1
set SYSGEN_MSXML_HTTP=1
set SYSGEN_NETUTILS=1
set SYSGEN_NOTIFY=1
set SYSGEN_PM=1
set SYSGEN_REDIR=1
set SYSGEN_SHELL=1
set SYSGEN_STANDARDSHELL=1
set SYSGEN_TOUCH=1
set SYSGEN_USB=1
set SYSGEN_USB_HID=1
set SYSGEN_USB_HID_CLIENTS=1
set SYSGEN_WININET=1
set SYSGEN_WININET_TWEENER=1
set SYSGEN_DOTNETV2=1
set SYSGEN_DOTNETV2_SUPPORT=1
set SYSGEN_CACHEFILT=1
set SYSGEN_IESAMPLE=1
set SYSGEN_IE_VBSCRIPT=1
set SYSGEN_LOCALAUDIO=1
set SYSGEN_STREAMAUDIO=1
set SYSGEN_STREAMAV=1
set SYSGEN_PWORD=1
set SYSGEN_TCPIP6=1
set SYSGEN_ETHERNET=1
set SYSGEN_WCELOAD=1
set BSP_NIC_RTL8139=1
set SYSGEN_ATAPI_PCIO=1
set SYSGEN_ATADISK=1
set SYSGEN_ATAPI_PCIO_CD=1
set SYSGEN_ATAPI=1
set SYSGEN_ATAPI_PCIP_PDC20262=1
set SYSGEN_DOTNETV2_SR=1
set SYSGEN_DOTNETV2_SR_CHS=1
set SYSGEN_PARALLEL=1
set SYSGEN_USB_STORAGE=1
set SYSGEN_USB_PRINTER=1
set SYSGEN_FSRAMROM=1
set SYSGEN_FSREGRAM=1
set SYSGEN_STOREMGR=1
set SYSGEN_UDFS=1
set SYSGEN_FATFS=1
set SYSGEN_MSPART=1
set SYSGEN_FONTS_ARIAL_1_30=1
set SYSGEN_MSPY3_DB_7=1
```

```
    set SYSGEN_SPIM_MSPY_LARGE=1
    set SYSGEN_FONTS_SIMSUN=1
    set SYSGEN_D3DMREF=1
    set SYSGEN_USBFN_STORAGE=1

    REM BSP features


    REM Misc settings
===============================================================================

    set WINCEDEBUG=retail
    set PATH=%PATH%;C:\Windows\system32;C:\Windows;C:\Program Files\Microsoft
Platform Builder\6.00\cepb\IdeVS

    REM Configuration environment variables

    set imgram128=1

    REM Build options

    set IMGNODEBUGGER=1
    set IMGNOTALLKMODE=1
    set IMGRAM64=
    set IMGNOKITL=1
    set ri_suppress_info=all
    set BUILD_MULTIPROCESSOR=1

    REM Project settings

    set
_USER_SYSGEN_BAT_FILES=C:\WINCE600\OSDesigns\OSDesign2\OSDesign2\Wince600\CEPC
_x86\OAK\MISC\OSDesign2.bat

    REM Locale options

    set IMGNOLOC=0
    set IMGSTRICTLOC=0
```

驱动程序组件的环境变量设置如下：

```
    @REM
    @REM Copyright (c) Microsoft Corporation.  All rights reserved.
    @REM
    @REM
    @REM Use of this sample source code is subject to the terms of the Microsoft
    @REM license agreement under which you licensed this sample source code. If
    @REM you did not accept the terms of the license agreement, you are not
    @REM authorized to use this sample source code. For the terms of the license,
    @REM please see the license agreement between you and Microsoft or, if
applicable,
    @REM see the LICENSE.RTF on your install media or the root of your tools
```

```
installation.
   @REM THE SAMPLE SOURCE CODE IS PROVIDED "AS IS", WITH NO WARRANTIES.
   @REM
   if /i not "%1"=="preproc" goto :Not_Preproc
       goto :EOF
   :Not_Preproc
   if /i not "%1"=="pass1" goto :Not_Pass1

       REM
===============================================================================
================
       REM
       REM Standard SDK features
       REM Post CE 5.0 this no longer supported,
       REM but is included for backwards compatibility.
       REM
===============================================================================
================
       if not "%SYSGEN_USDK%"=="1" goto NoUSDK
           set SYSGEN_AYGSHELL=1
           set SYSGEN_AUDIO=1
           set SYSGEN_WININET=1
           set SYSGEN_URLMON=1
           set SYSGEN_CPP_EH_AND_RTTI=1
           set SYSGEN_REDIR=1
           set SYSGEN_MSXML_DOM=1
           set SYSGEN_ATL=1
           set SYSGEN_SOAPTK_CLIENT=1
           set SYSGEN_MSMQ=1
           set SYSGEN_LDAP=1
           set SYSGEN_OBEX_CLIENT=1
           set SYSGEN_AUTH=1
           set SYSGEN_GRADFILL=1
           set SYSGEN_PRINTING=1
           set __SYSGEN_STANSDK=1
           set __SYSGEN_COM_GUIDS=1
           set __SYSGEN_COM_STG=1
           set SYSGEN_COMMDLG=1
           set SYSGEN_STDIOA=1
           set SYSGEN_STANDARDSHELL=1
           set SYSGEN_MSXML_SAX=1
           set SYSGEN_DOTNETV2_SUPPORT=1
       :NoUSDK

       goto :EOF
   :Not_Pass1
   if /i not "%1"=="pass2" goto :Not_Pass2
       goto :EOF
   :Not_Pass2
   if /i not "%1"=="report" goto :Not_Report
       goto :EOF
   :Not_Report
```

```
echo %0 Invalid parameter %1
```

为了使系统编译成中文，系统可以快速正常启动，项目的属性设置如图 2-16 所示，项目的编译选项设置如图 2-17 所示，添加的环境变量如图 2-18 所示。

最后，为了使编译成功的操作系统可以在 VMWare 虚拟机中使用网络，需要加入 VMWare 虚拟机的网卡驱动以及设置注册表。

该驱动程序可以在 AMD 公司的网站上下载。

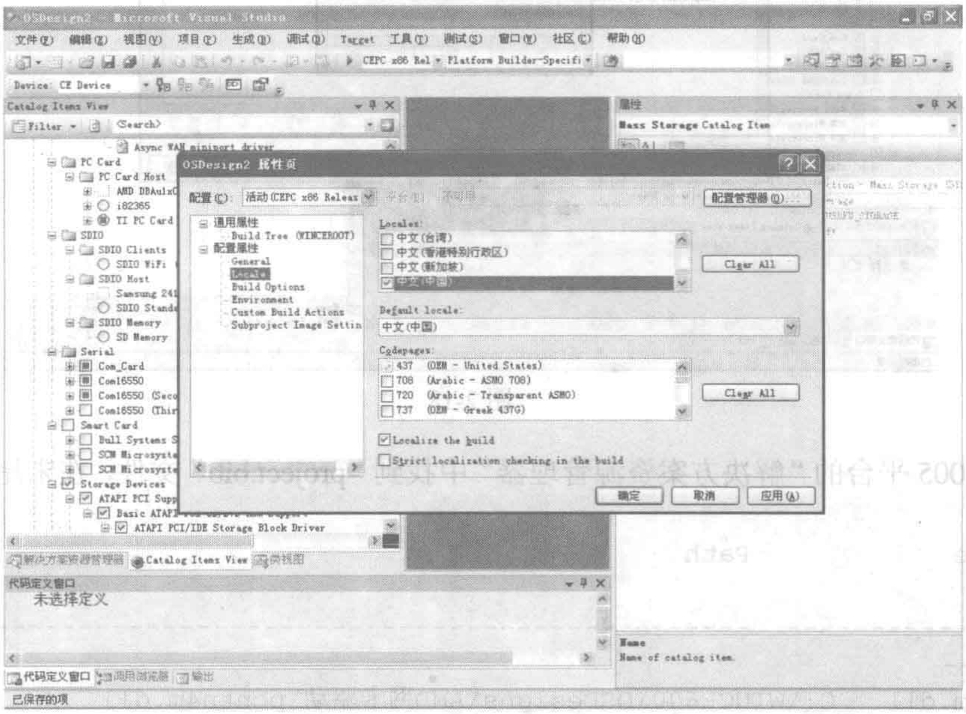


图 2-16

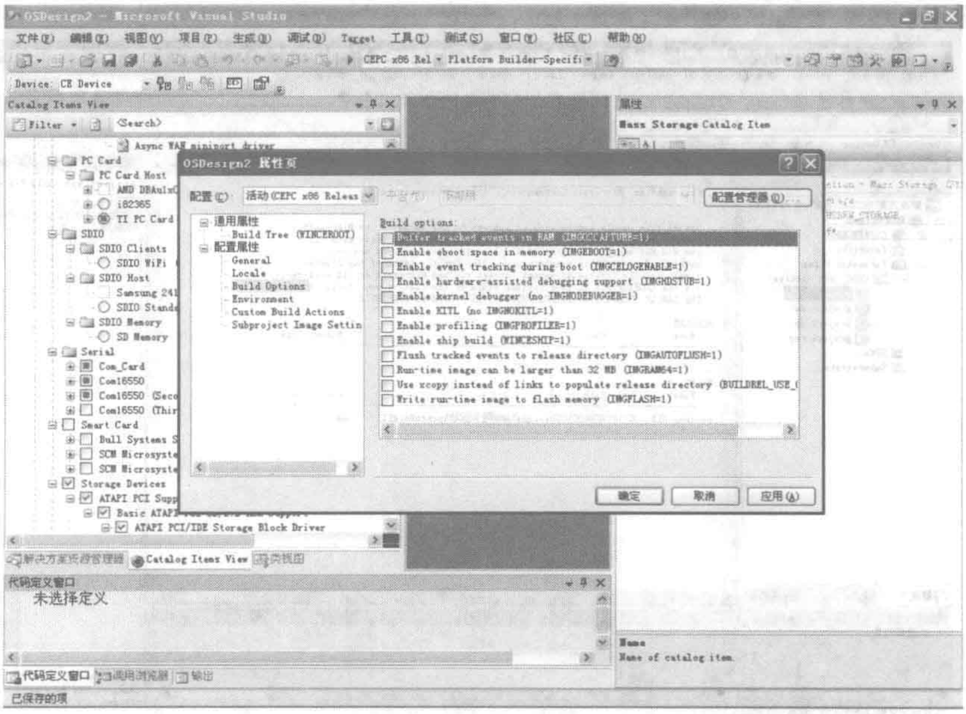


图 2-17

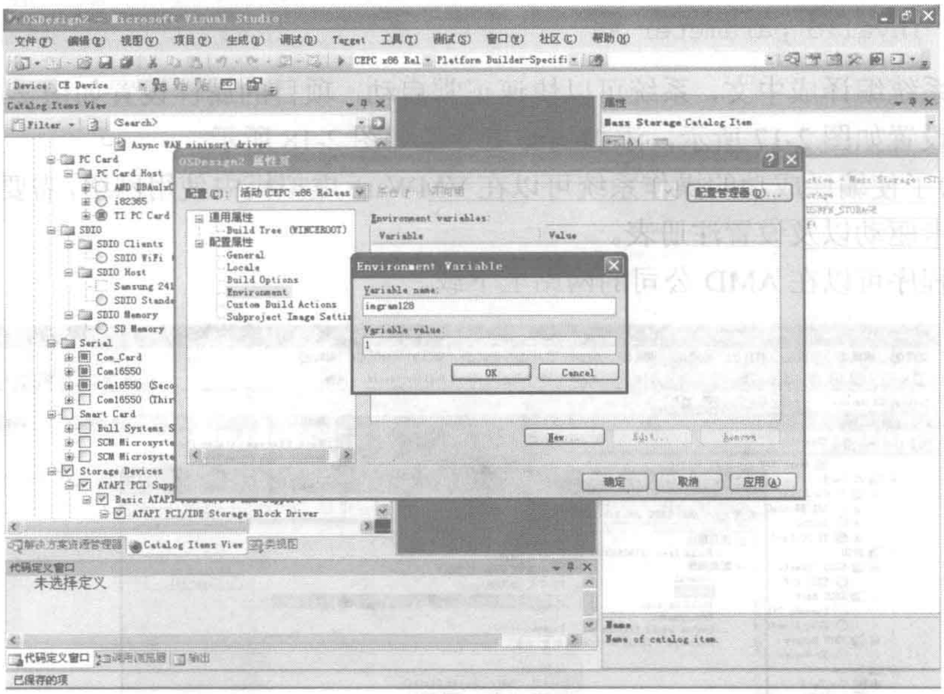


图 2-18

在 VS2005 平台的“解决方案资源管理器”中找到“project.bib”文件，在末尾加入如下行。

```
    FILES
    ;  Name               Path                                                    Memory
Type
    ;  ---------------    ----------------------------------------------------
-----------
    pcntn4m.dll    C:\WINCE600\OSDesigns\AMD网卡驱动\pcntn4m.dll       nk
```

这样，就可以将 VMWare 虚拟机的网卡驱动程序加入编译成的操作系统 nk.bin 文件当中，如图 2-19 所示。

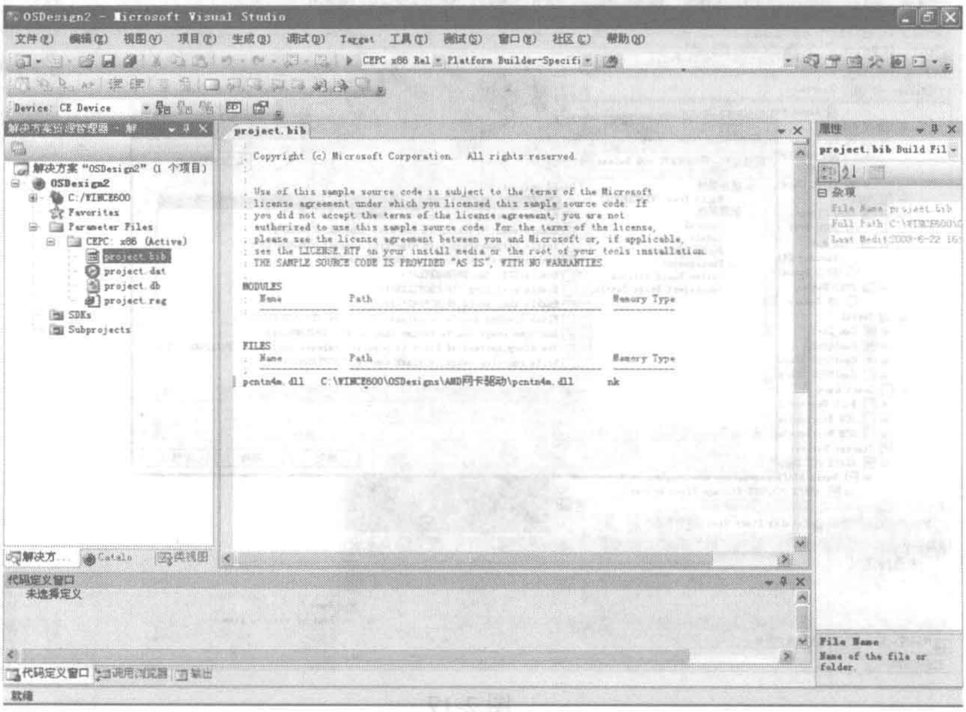


图 2-19

最后，需要设置注册表，让驱动程序生效。如图 2-20 所示。在 VS2005 平台的“解决方案资源管理器”中找到“project.reg”文件，切换到“Source”标签页，在末尾加入如下行：

```
[HKEY_LOCAL_MACHINECommPCNTN4M]
    "DisplayName"="PCNTN4M Compatible Ethernet Driver"
    "Group"="NDIS"
    "ImagePath"="pcntn4m.dll"

[HKEY_LOCAL_MACHINECommPCNTN4MLinkage]
    "Route"=multi_sz:"PCNTN4M1"

[HKEY_LOCAL_MACHINECommPCNTN4M1]
    "DisplayName"="PCNTN4M Compatible Ethernet Driver"
    "Group"="NDIS"
    "ImagePath"="pcntn4m.dll"


[HKEY_LOCAL_MACHINECommPCNTN4M1Parms]
;     "BusNumber"=dword:0
;     "BusType"=dword:05
;     "Interrupt"=dword:05
;     "IOAddress"=dword:0300


[HKEY_LOCAL_MACHINECommTcpipLinkage]
       "Bind"=multi_sz:"ppp","PCNTN4M1"

    ; Registry values for the pcntn4m1 driver
[HKEY_LOCAL_MACHINECommPCNTN4M1ParmsTcpIp]
    ; This enable the DHCP. In Win CE 2.1 Preview version statically allocated
IP address
        ; does not work. So we use the DHCP server to allocate the IP address.
    "EnableDHCP"=dword:1
    ; This should be MULTI_SZ
    "DefaultGateway"=""
    ; This should be SZ... If null it means use LAN, else WAN and Interface.
    "LLInterface"=""
    ; Use zero for broadcast address? (or 255.255.255.255)
    "UseZeroBroadcast"=dword:0
    ; Thus should be MULTI_SZ, the IP address list
    "IpAddress"="0.0.0.0"
    ; This should be MULTI_SZ, the subnet masks for the above IP addresses
    "Subnetmask"="0.0.0.0"


;
; Template the PCI bus driver uses to match a AM79C970 PC-net card
;
[HKEY_LOCAL_MACHINEDriversBuiltInPCITemplatePCNTN4M]
   "Dll"="NDIS.dll"
   "Class"=dword:02
   "SubClass"=dword:00
   "ProgIF"=dword:0
```

```
    "VendorID"=multi_sz:"1022"
    "DeviceID"=multi_sz:"2000"
;   "Entry"="NdisPCIBusDeviceInit"
    "Prefix"="NDS"
    "Transceiver"=dword:3
    "IsrDll"="giisr.dll"
    "IsrHandler"="ISRHandler"
    "PortIsIO"=dword:1
    "PortOffset"=dword:C          ; TBD
    "PortSize"=dword:2
    "PortMask"=dword:5F00     ; TBD

[HKEY_LOCAL_MACHINECommConnectionSharing]
    "PrivateInterface"="PCI\PCNTN4M"
```

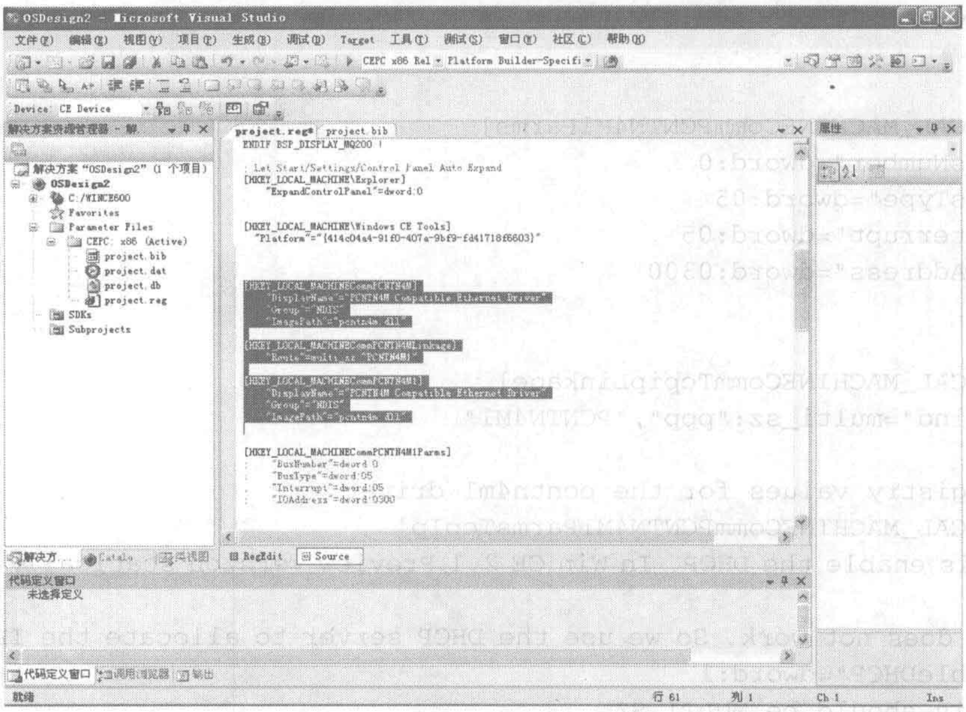


图 2-20

这样，最后编译出来的操作系统就可以在 VMWare 虚拟机中运行并成功使用包括声卡、显卡、网卡等在内的绝大部分硬件。系统运行效果请参考 4.1 节。值得注意的是，第一次编译系统可能会花费较长的时间，这同开发平台的硬件配置有关。尤其是在开启杀毒软件的实时监控时，所花费的时间更会成倍增加。并且，往往不到编译的最后，是无法预知当前系统正确与否的。

第3章　运行 Windows CE 系统

经过漫长的操作系统安装、开发平台安装、WINCE 操作系统定制，直至最终的 NK.bin 文件编译成功，我们终于可以看到 WINCE 系统运行的效果了。如果没有虚拟机平台，很多人可能直接放弃了学习 WINCE，因为寻找一款完全适合 WINCE 自带的 BSP 标准配置的硬件平台并不是每一个人都可以做到的。实际上，VMWare 虚拟机的各个主要硬件，包括声卡、显卡、网卡、USB 接口等都可以被 WINCE 驱动成功。另外，大部分的 PC 机也可以运行 WINCE，当然不排除部分硬件不能正确驱动。

3.1　配置 VMWare 运行环境

首先，安装完成 VMWare 软件之后，按照下图步骤建立一个虚拟机，配置好系统运行环境。

步骤 1 如图 3-1 所示，新建一个虚拟机，打开 VMWare 软件，选择“File”→“New”→“Virtual Machine”选项。

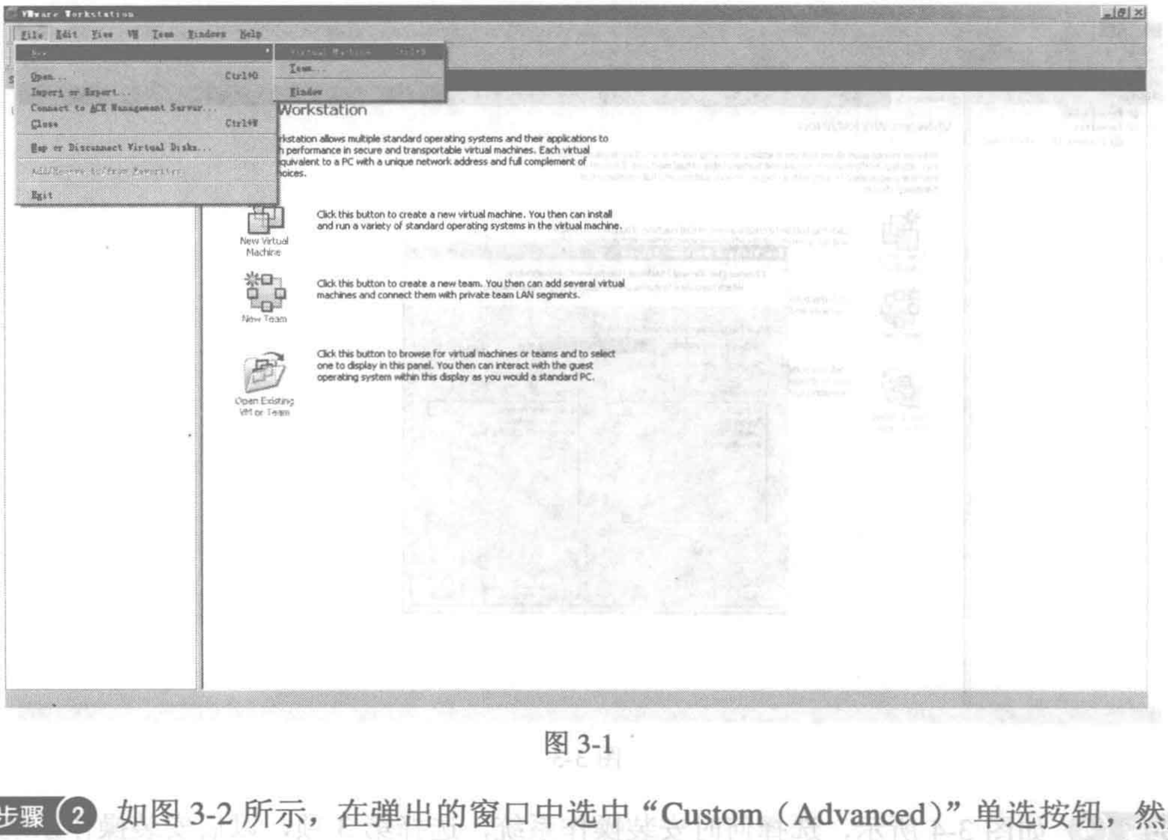


图 3-1

步骤 2 如图 3-2 所示，在弹出的窗口中选中“Custom（Advanced）”单选按钮，然后单击“Next”按钮。

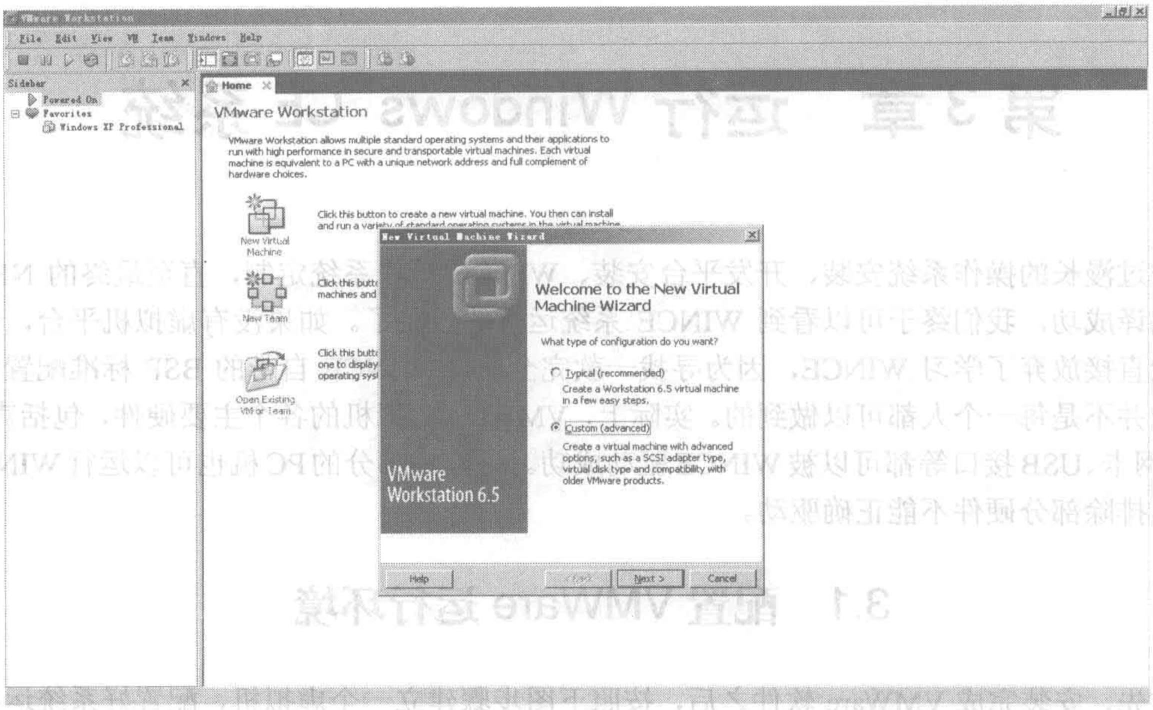


图 3-2

步骤 3 如图 3-3 所示，选择 VMWare 版本号，选择 6.0 或者 6.5 版本，然后单击“Next”按钮。

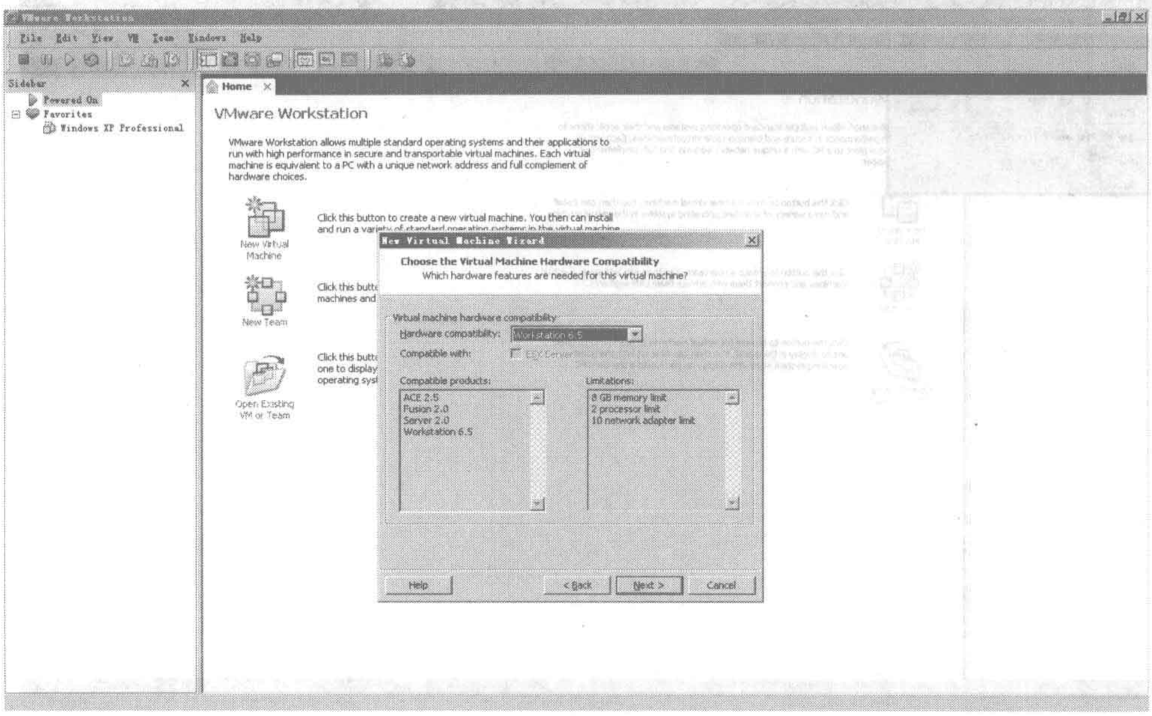


图 3-3

步骤 4 如图 3-4 所示，选择何时安装操作系统，选择第 3 项，以后安装操作系统，然后单击“Next”按钮。

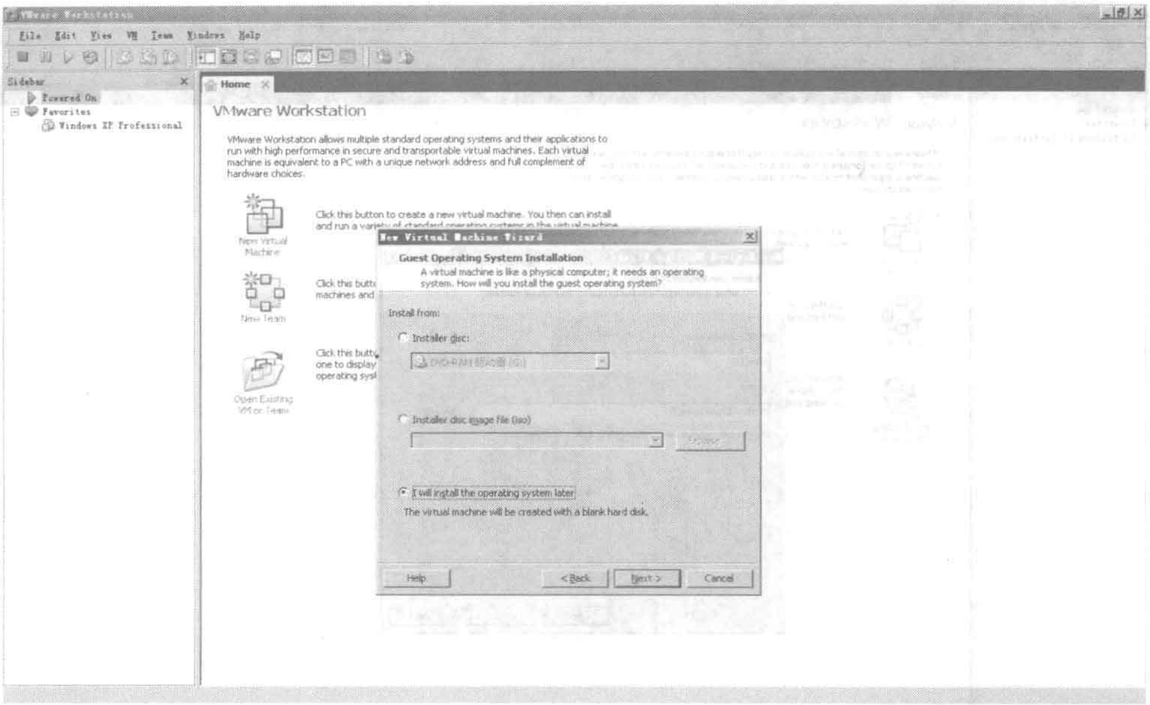


图 3-4

步骤 5 如图 3-5 所示，选择操作系统类型，这里请选择微软的“Windows 98”，然后单击“Next”按钮。

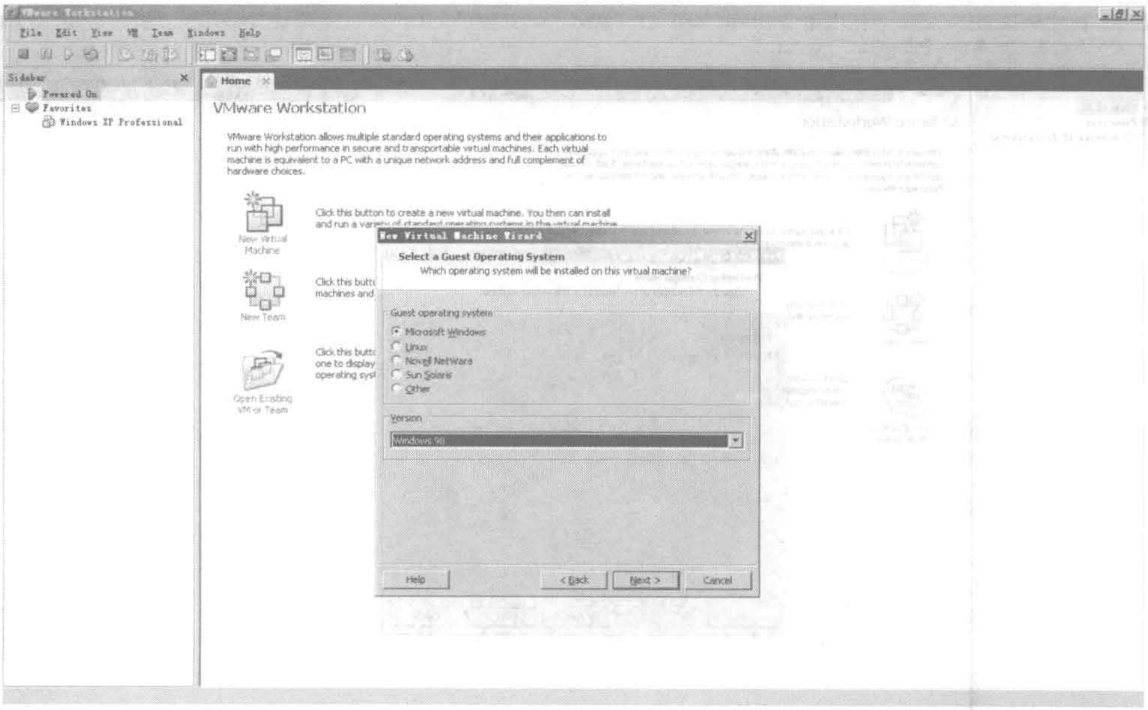


图 3-5

步骤 6 如图 3-6 所示，给虚拟机命名，可以任意填写，建议写成 WINCE6，选择虚拟机配置保存的位置，请自己规划好文件保存的路径，本例保存在 f:\WINCEBook\WINCE6 目录下，然后单击“Next”按钮。

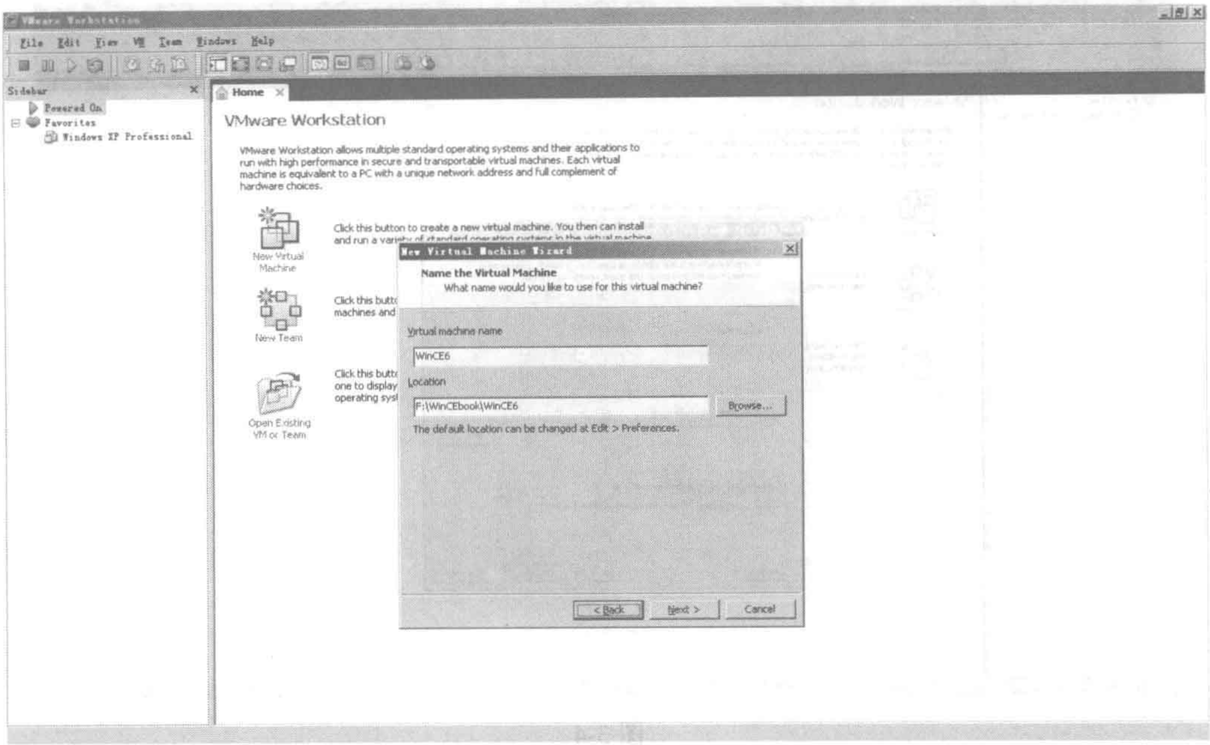


图 3-6

步骤 7 如图 3-7 所示，选择处理器个数，请选择一个，然后单击“Next”按钮。

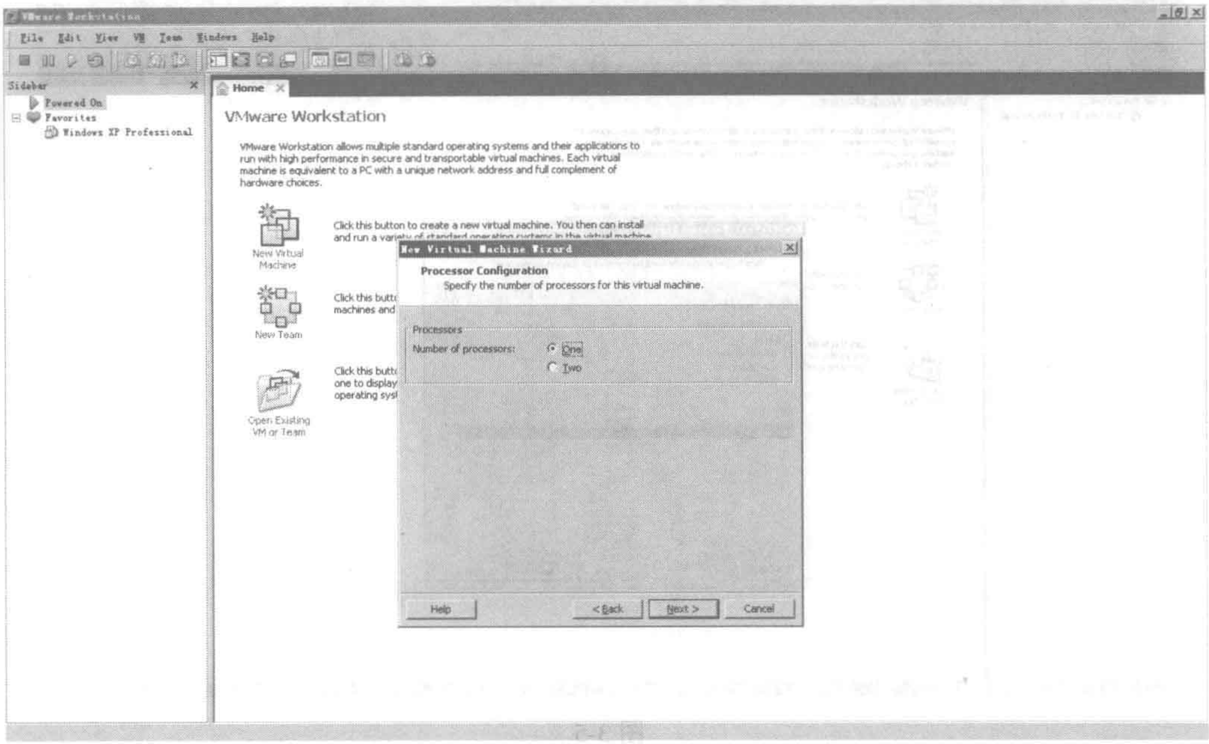


图 3-7

步骤 8 如图 3-8 所示，设置内存大小。根据编译 WINCE 系统编译的参数设置，最小应该设置为 32 MB，最大不超过 512 MB。若内存足够，设置成 512 MB 最佳，确保 WINCE 系统启动时不会发生内存不足的错误。本例设置成 64 MB，然后单击“Next”按钮。

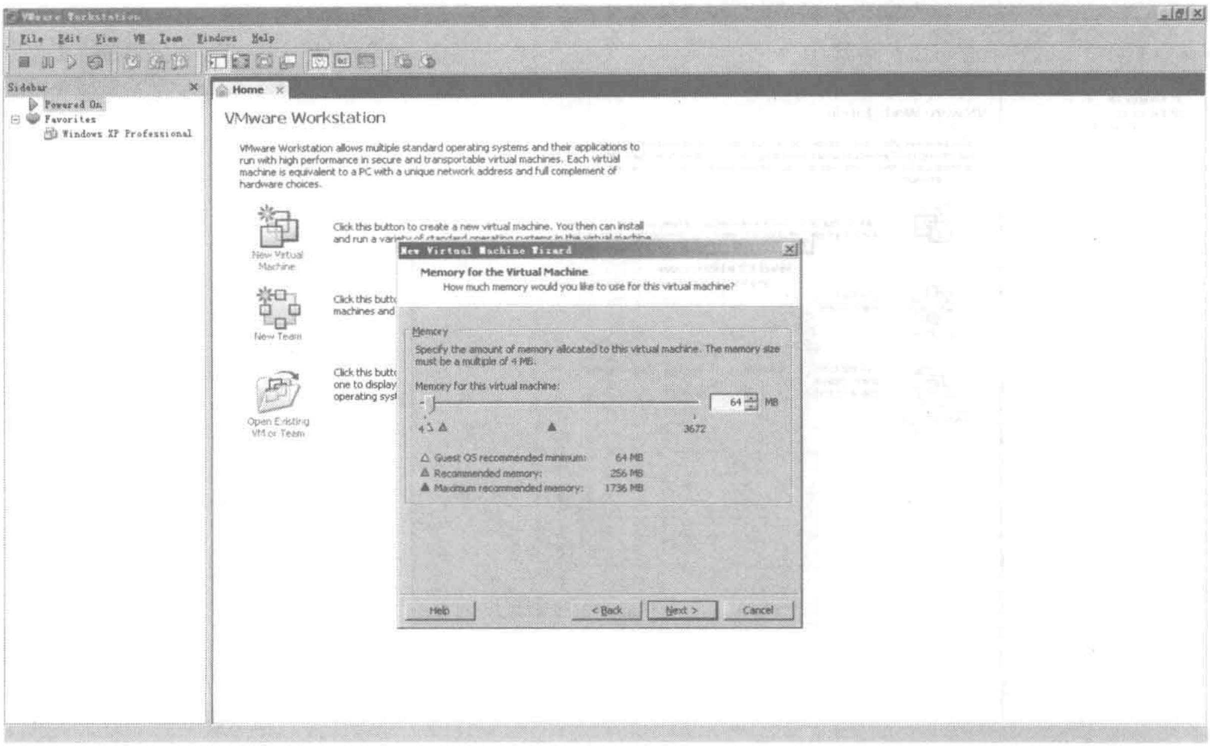


图 3-8

步骤 9 如图 3-9 所示，选择网络连接类型。请选择 NAT 方式，然后单击“Next”按钮。

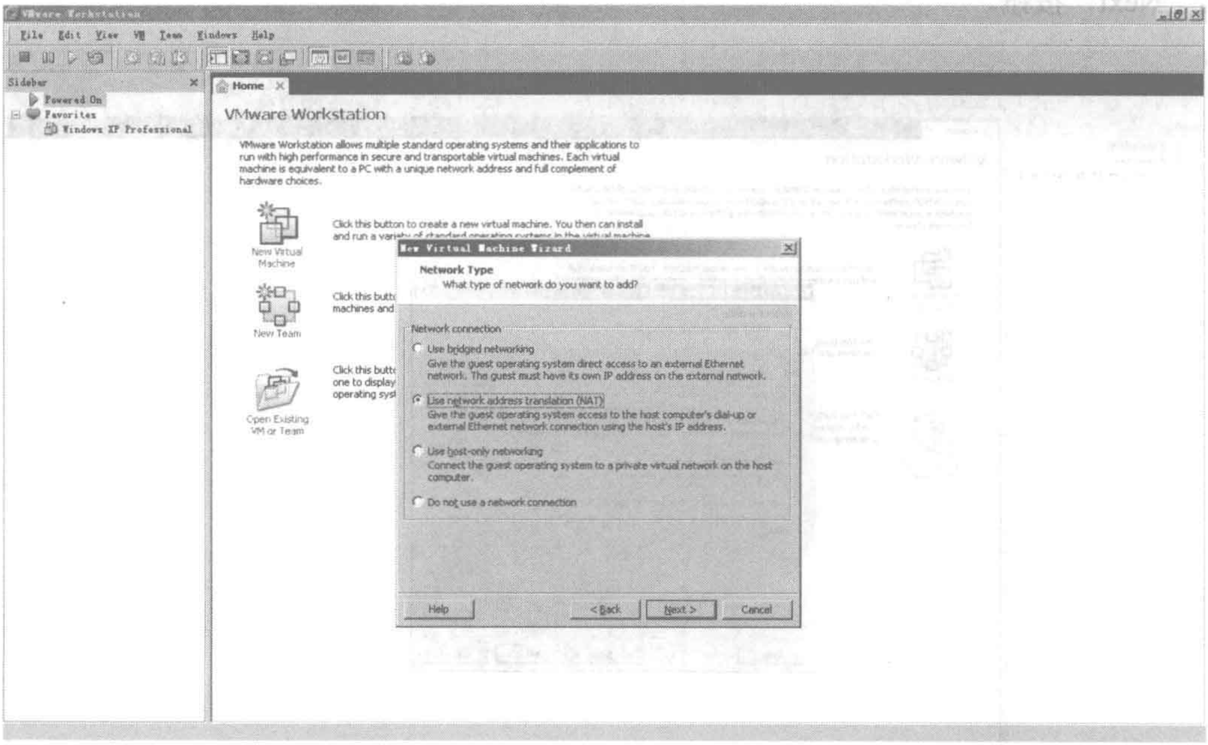


图 3-9

步骤 10 如图 3-10 所示，选择虚拟磁盘的工作方式，按默认方式选择，然后单击“Next”按钮。

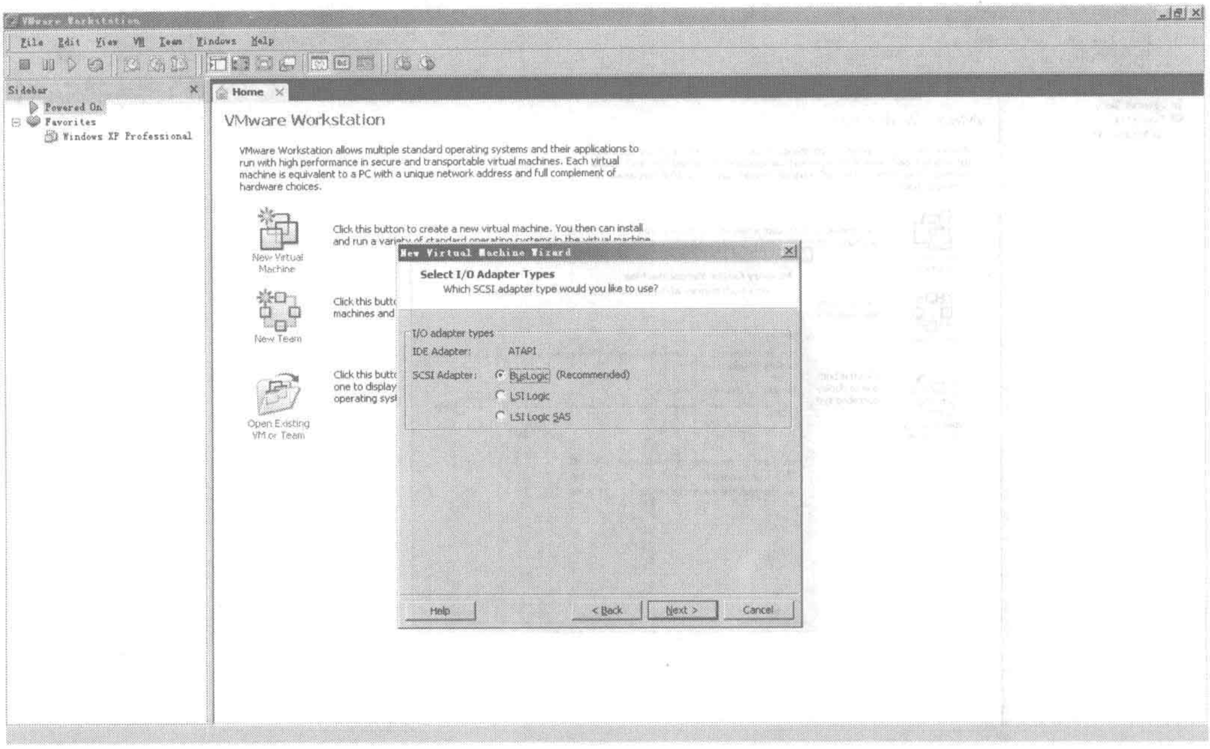


图 3-10

步骤 11 如图 3-11 所示，创建虚拟磁盘。请选择第一项，创建一个新的虚拟磁盘，然后单击“Next”按钮。

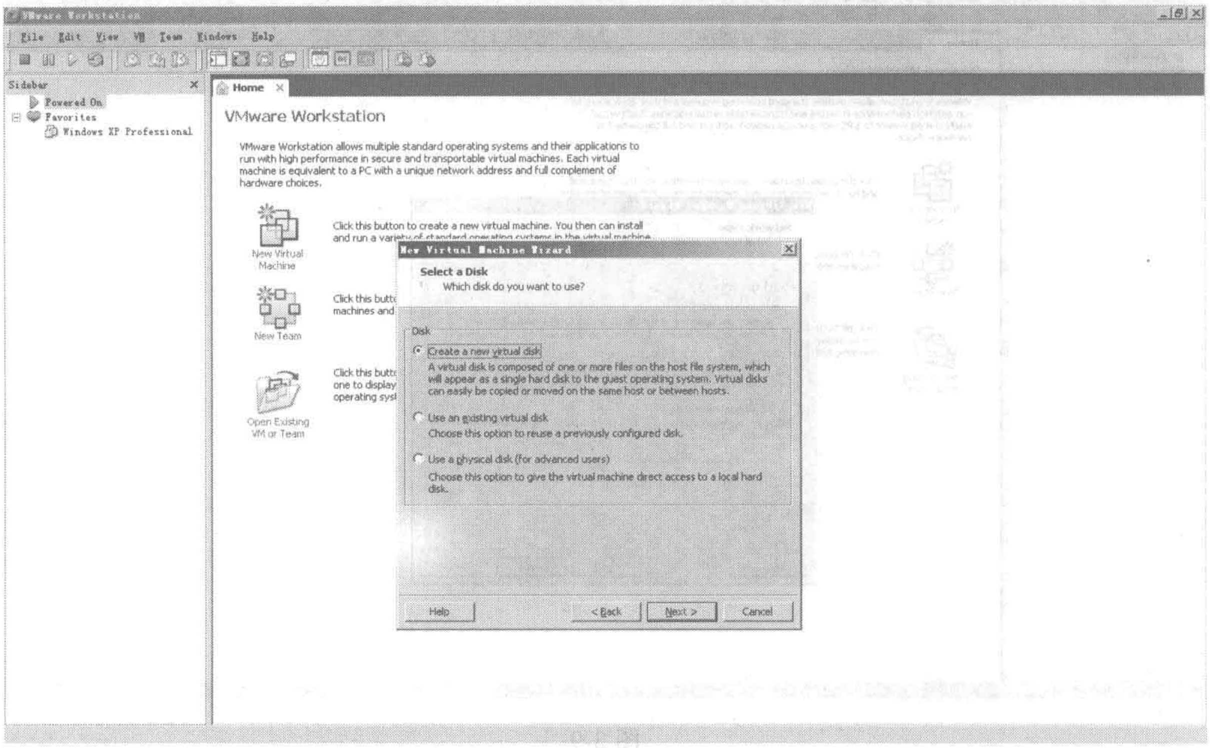


图 3-11

步骤 12 如图 3-12 所示，选择虚拟磁盘的接口方式，请选择 IDE 接口，若需要更高磁盘速度，请选择 SCSI 接口，然后单击“Next”按钮。

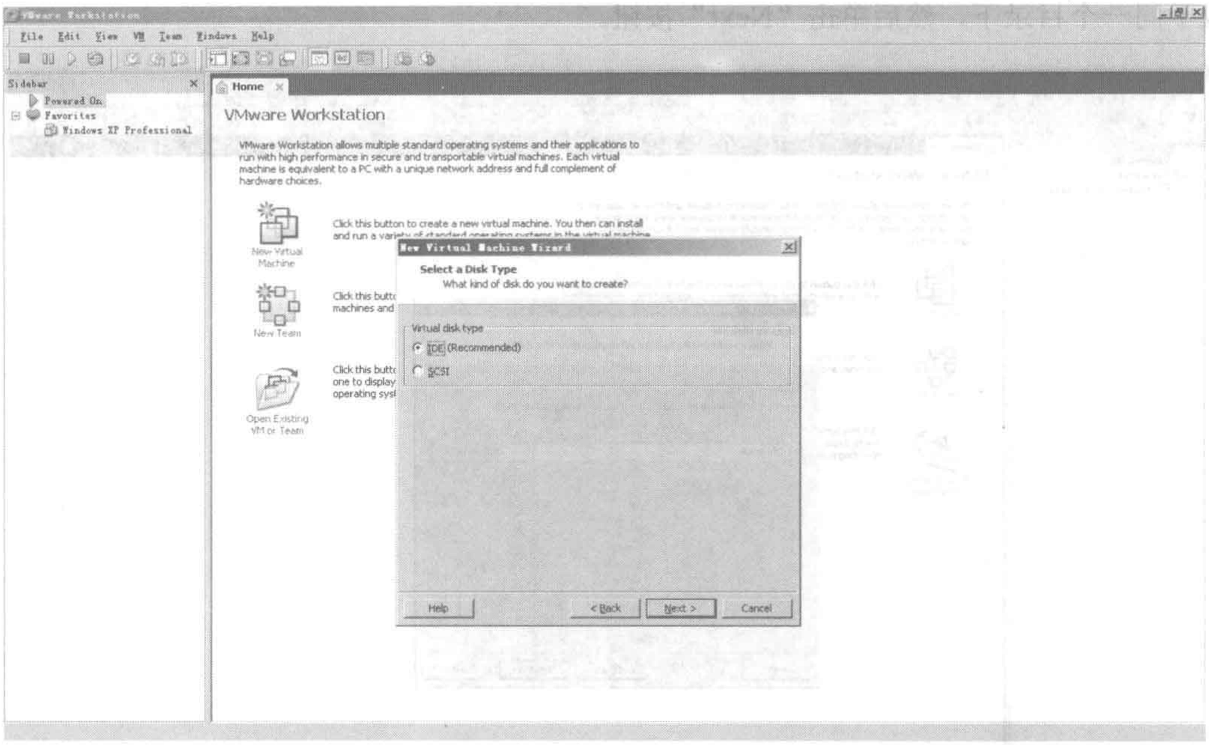


图 3-12

步骤 13 如图 3-13 所示，选择磁盘大小。由于 NK.bin 文件一般在 10 MB 到 64 MB 之间，硬盘上如果没有太多其他文件需要存放，可适当将硬盘设置小一点。但 VMWare 虚拟机的最小硬盘规定为 100 MB。这里可以根据自己的需求适当调整，然后单击“Next”按钮。

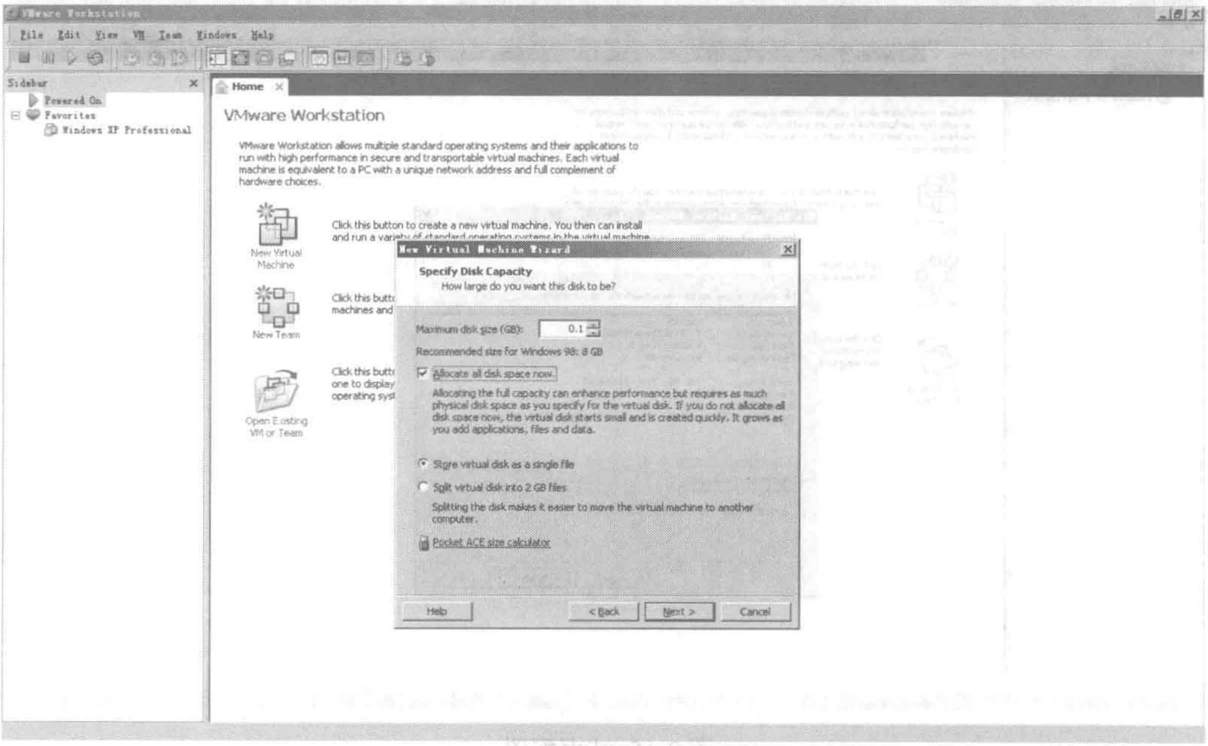


图 3-13

步骤 14 如图 3-14 所示，选择虚拟磁盘文件的存放位置，可自由规划，最好同配置文件

存放于同一个目录下，然后单击“Next”按钮。

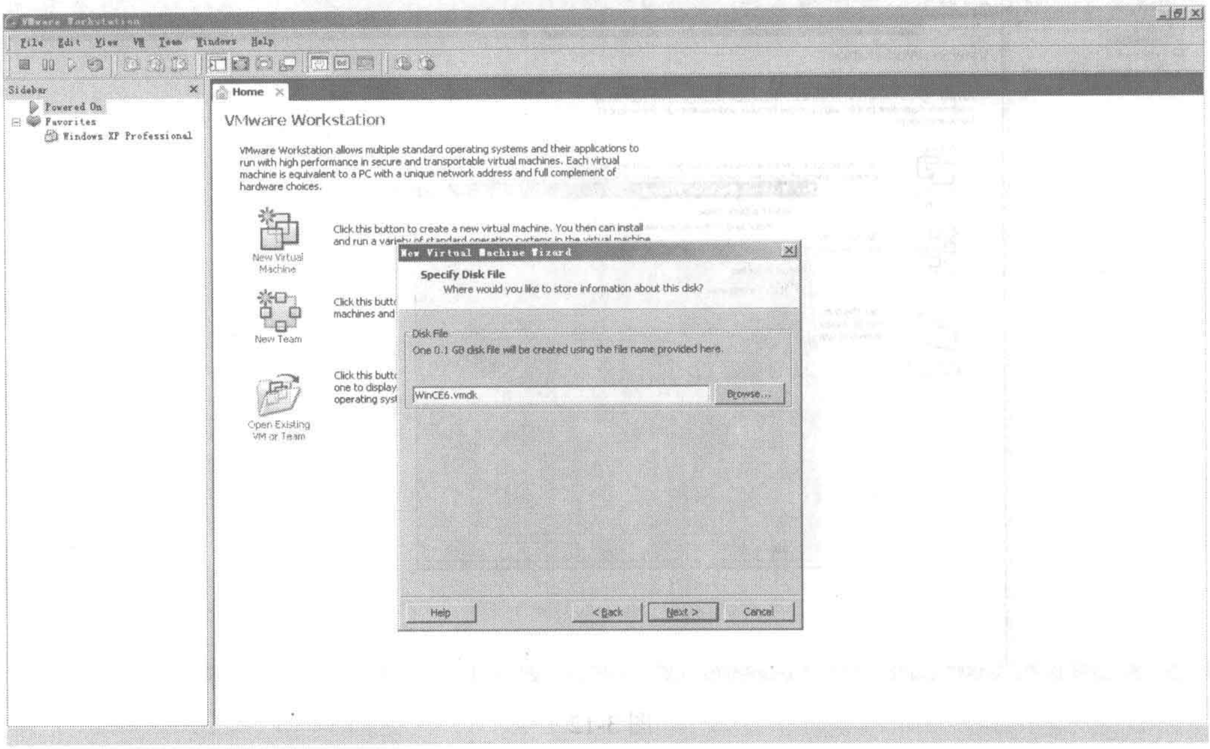


图 3-14

步骤 15 如图 3-15 所示，单击“Finish”按钮结束虚拟机的配置。

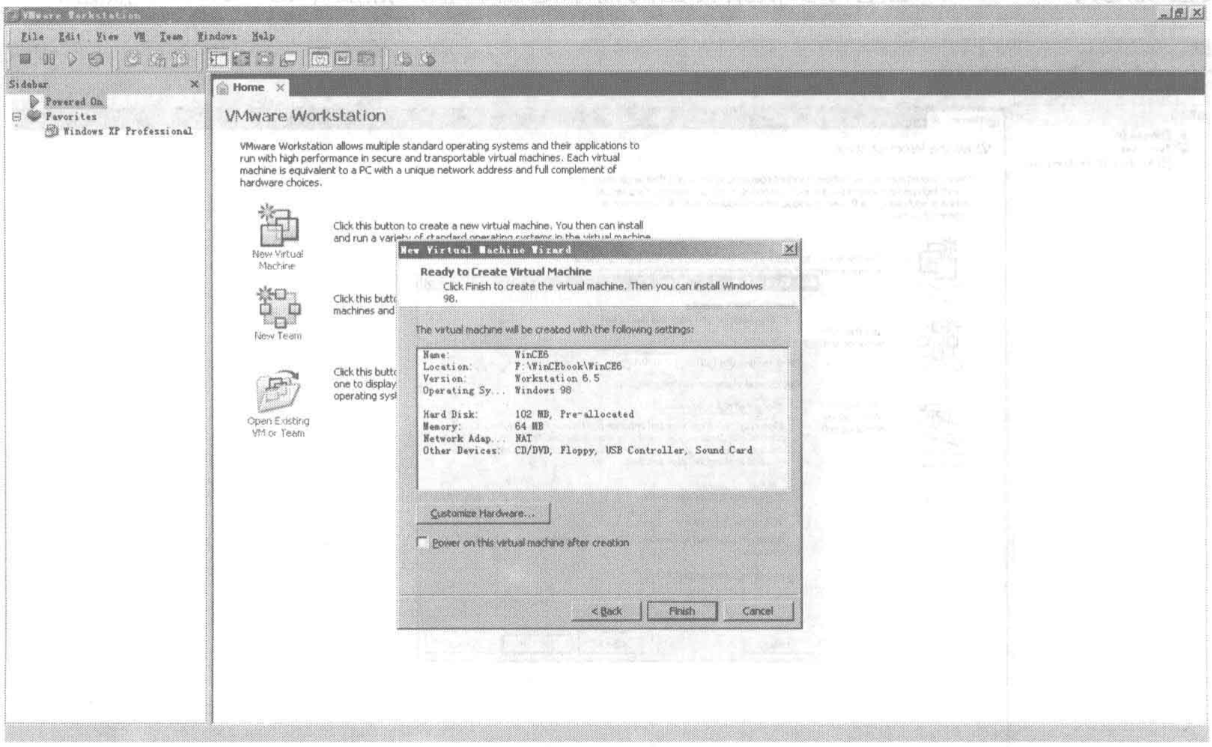


图 3-15 结束配置

配置完成之后，VMWare 虚拟机平台上将出现新添加的虚拟系统，如图 3-16 所示。

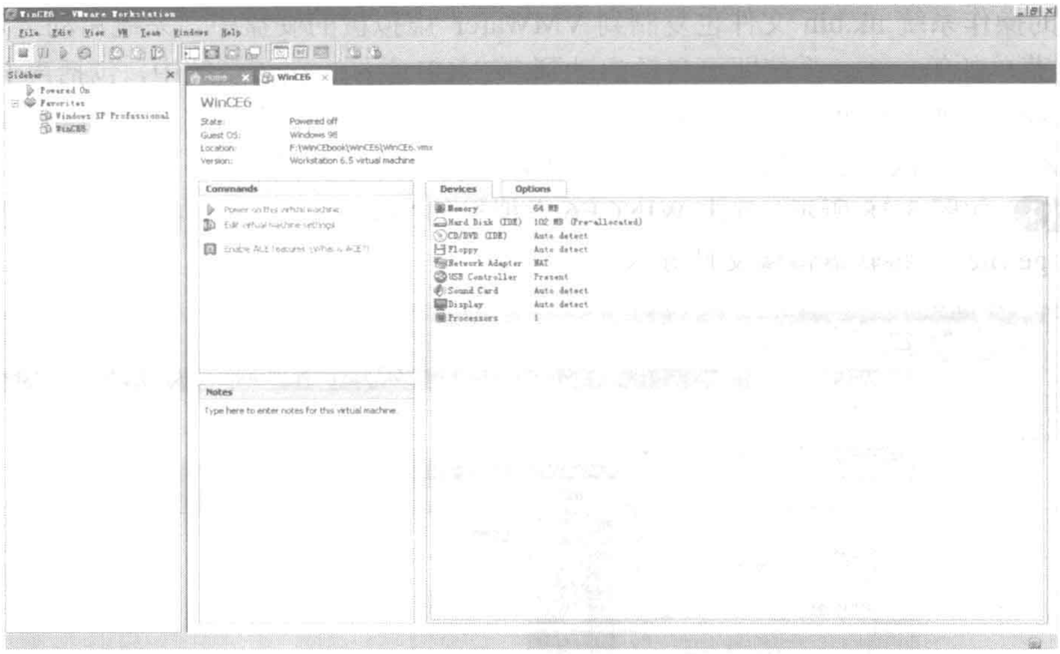


图 3-16

接下来，需要在 VMWare 虚拟机上建立 WINCE 的启动机制。这里采用 MSDOS 与 LoadCEPC 的方式来进行。首先，需要建立一张启动用的软盘。最好找一个带软驱的电脑，如果软驱不是太好找，也可以使用模拟软驱的 U 盘。在“C:\Program Files\Microsoft Platform Builder\6.00\cepb\utilities”目录下找到“MakeImageDisk.exe”文件，选择该目录下的“CepcBoot.144”文件，放入软件，制作启动盘，如图 3-17 所示。

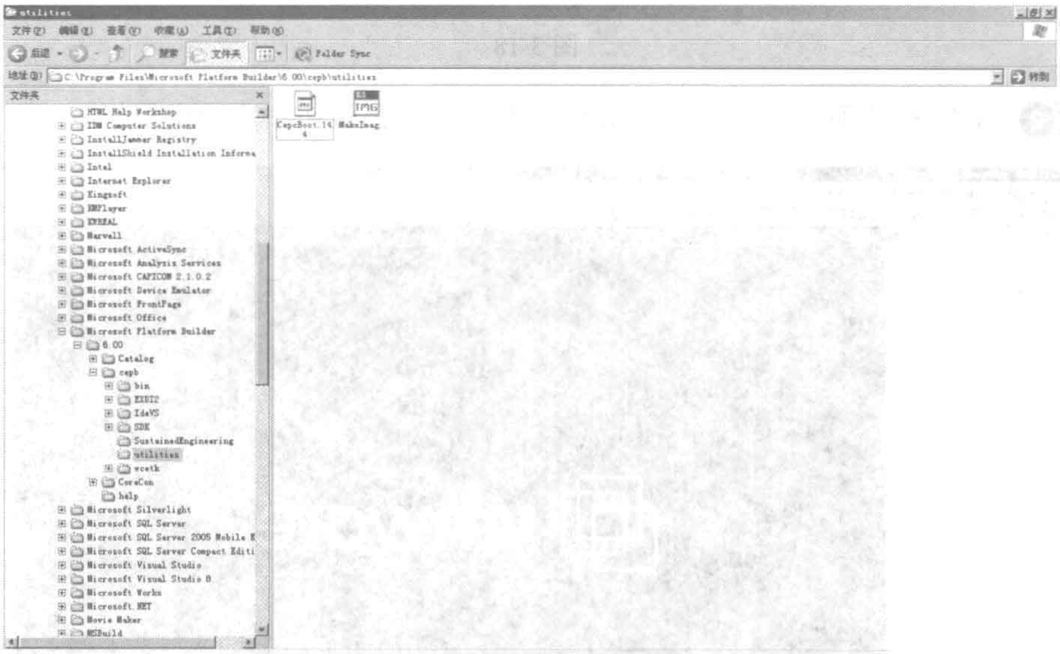


图 3-17

将该软件制作成软盘镜像文件，在 VMWarer 虚拟机的软驱设置当中选择软盘镜像文件启动。然后利用“fdisk.exe”和“format.exe”将 VMWarer 虚拟机的硬盘分区、格式化，制作成 DOS 系统。然后将软盘镜像上的所有文件都复制到 VMWarer 虚拟机的硬盘当中。最后，

将编译好的操作系统 nk.bin 文件也复制到 VMWarer 虚拟机的硬盘中。系统启动之后，选择 DOS 启动菜单的第一项，系统即可自动启动到 WINCE 系统。在该系统中，包括声卡、显卡、网卡等大部分的硬件都可以正常工作。这样，不需要特殊的硬件配置，也不需要特定的嵌入式主板，就在一台 PC 机上搭建了一个完整的 WINCE6 的开发平台。

步骤 1 如图 3-18 所示，双击 WINCE6 虚拟机的“Floppy”选项，在新窗口中选择“Use floppy image file”，将软盘镜像文件导入。

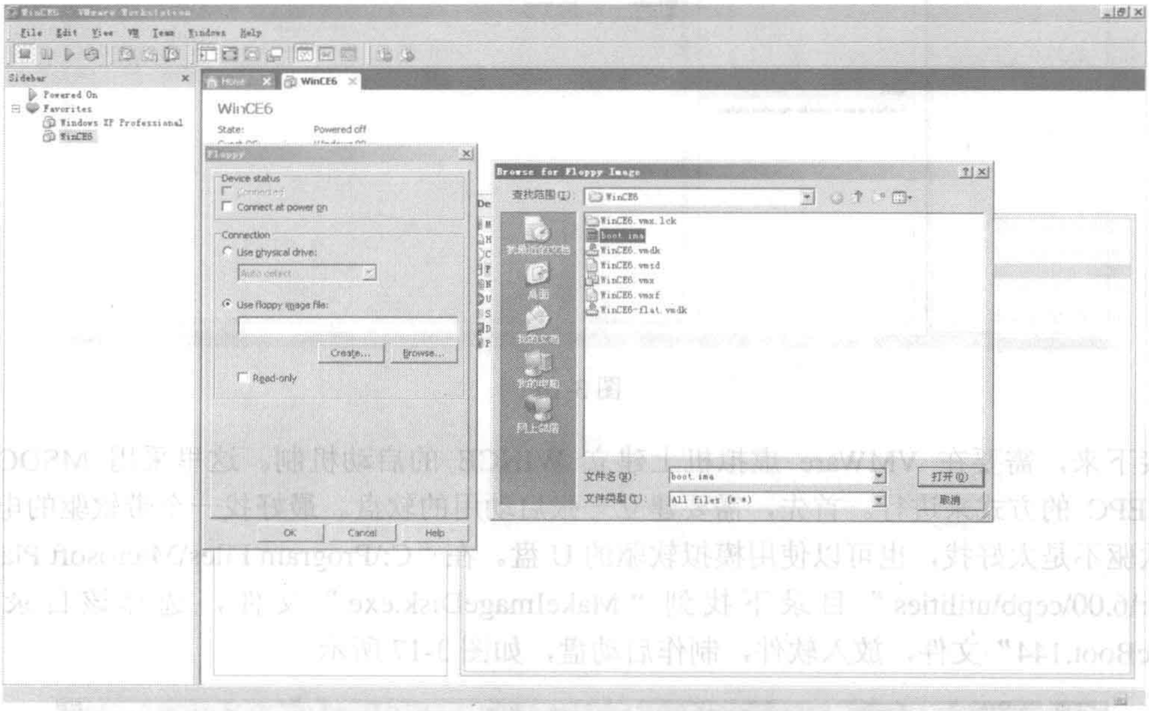


图 3-18

步骤 2 如图 3-19 所示，单击 VMWare 工具栏的绿色按钮，启动虚拟机。

图 3-19

步骤 3 如图 3-20 所示，软盘镜像引导成功之后，进入 DOS 系统。

图 3-20

步骤 4 如图 3-21 所示，在 DOS 系统下利用 Fdisk 对虚拟机硬盘进行分区，只需要划分一个分区，不建议划分多个分区。

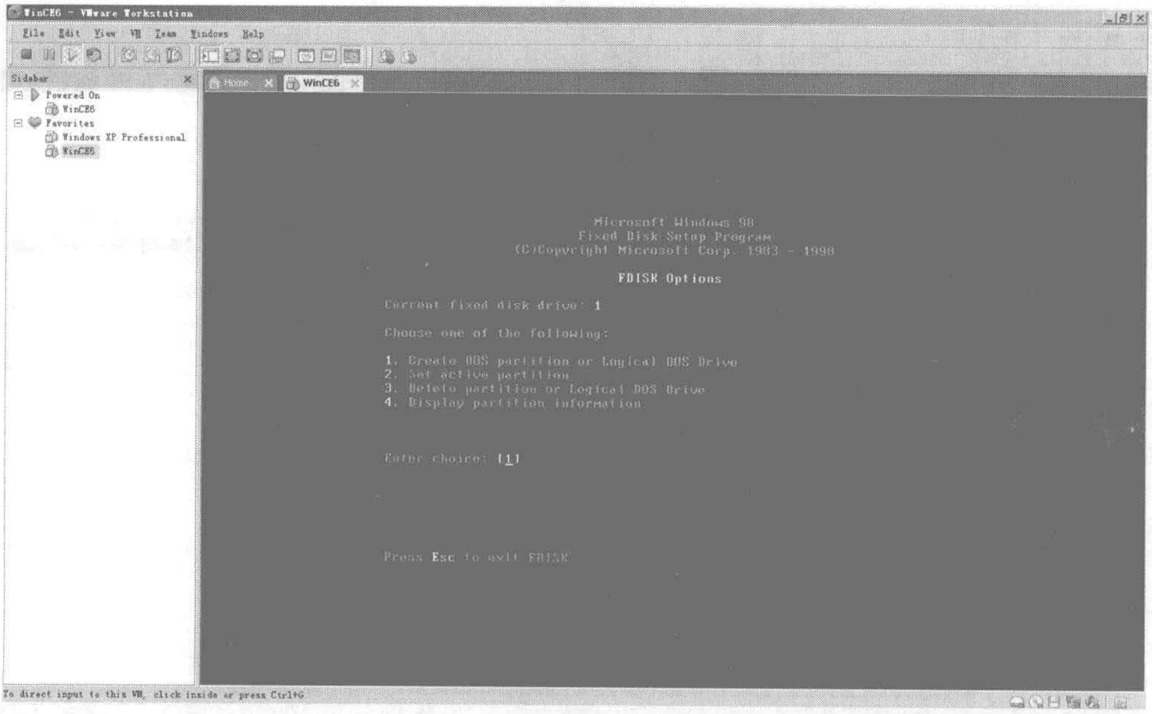


图 3-21

步骤 5 如图 3-22 所示，重启系统后，对虚拟机硬盘进行格式化，注意使用 format c: /s 参数，将硬盘格式化为带 DOS 系统。

图 3-22

步骤 6 如图 3-23 所示，复制文件到虚拟机硬盘。双击“Hard Disk（IDE)”选项，在弹出窗口中单击“Utilities”，然后再选择“Map…”选项，在弹出的窗口中将只读属性单选框清除。将虚拟机硬盘映射为本机的一个盘符。

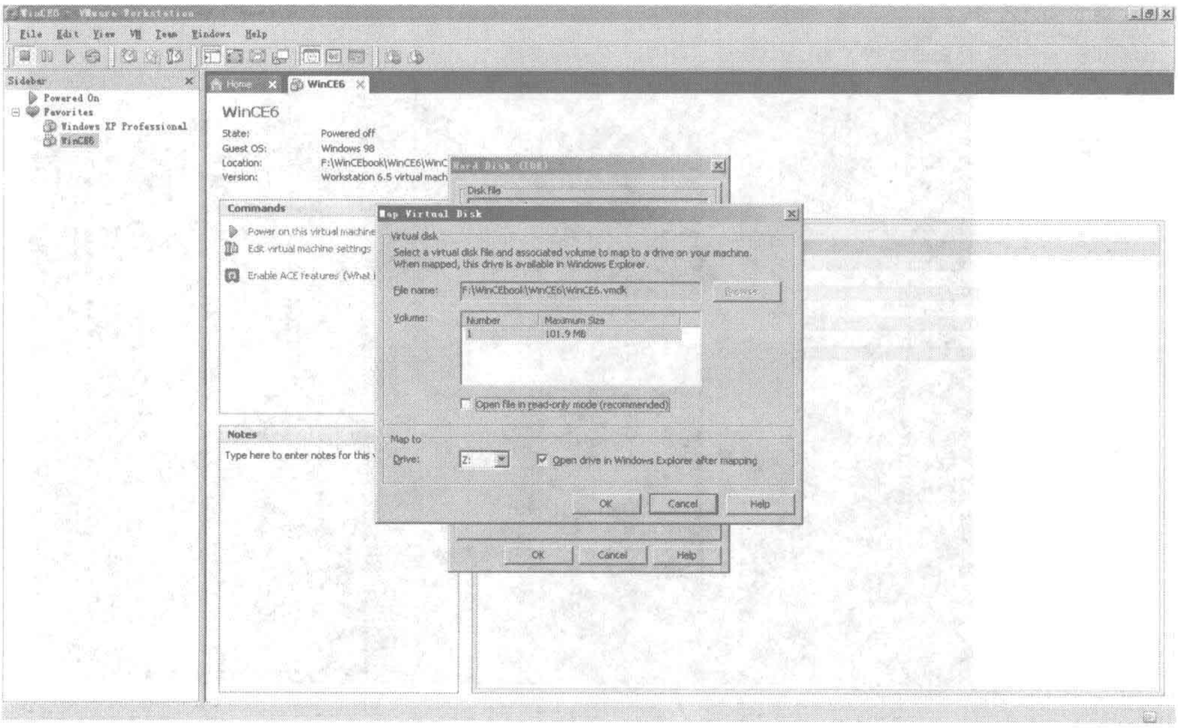


图 3-23

步骤 7 如图 3-24 所示，将虚拟软盘上的所有文件以及 NK.bin 文件复制到虚拟机硬盘。

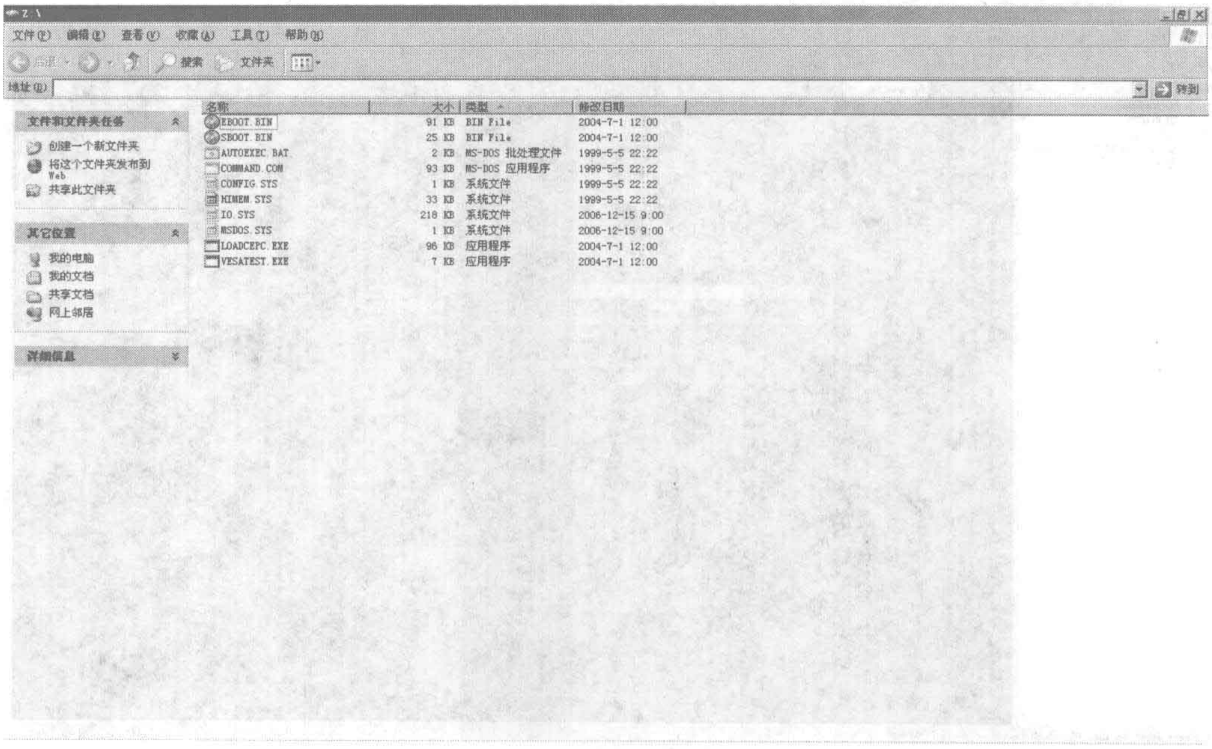


图 3-24

步骤 8 如图 3-25 所示，断开虚拟机到本机的磁盘映射。

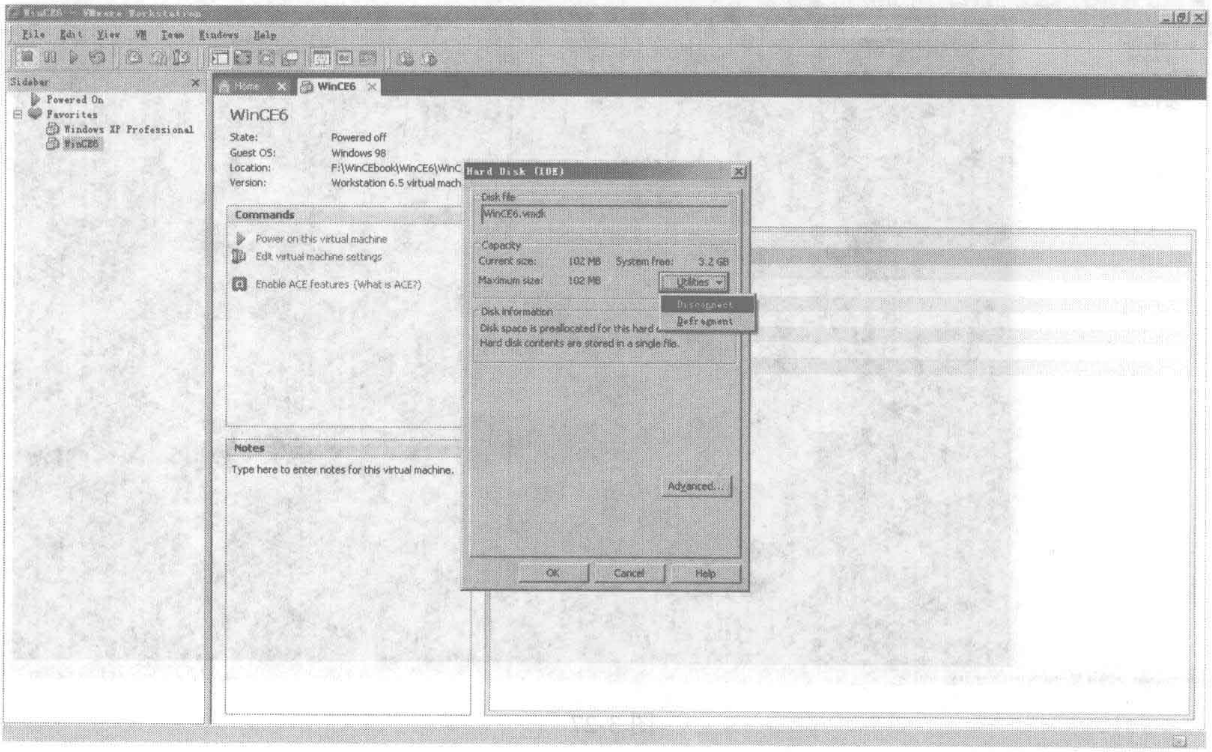


图 3-25

步骤 9 如图 3-26 所示，重新启动虚拟机，选择从本地启动 nk.bin 文件。根据硬件配置的不同，检测与加载的时间从几秒到 30 秒左右不等。加载过程有进度条显示，如图 3-27 所示。

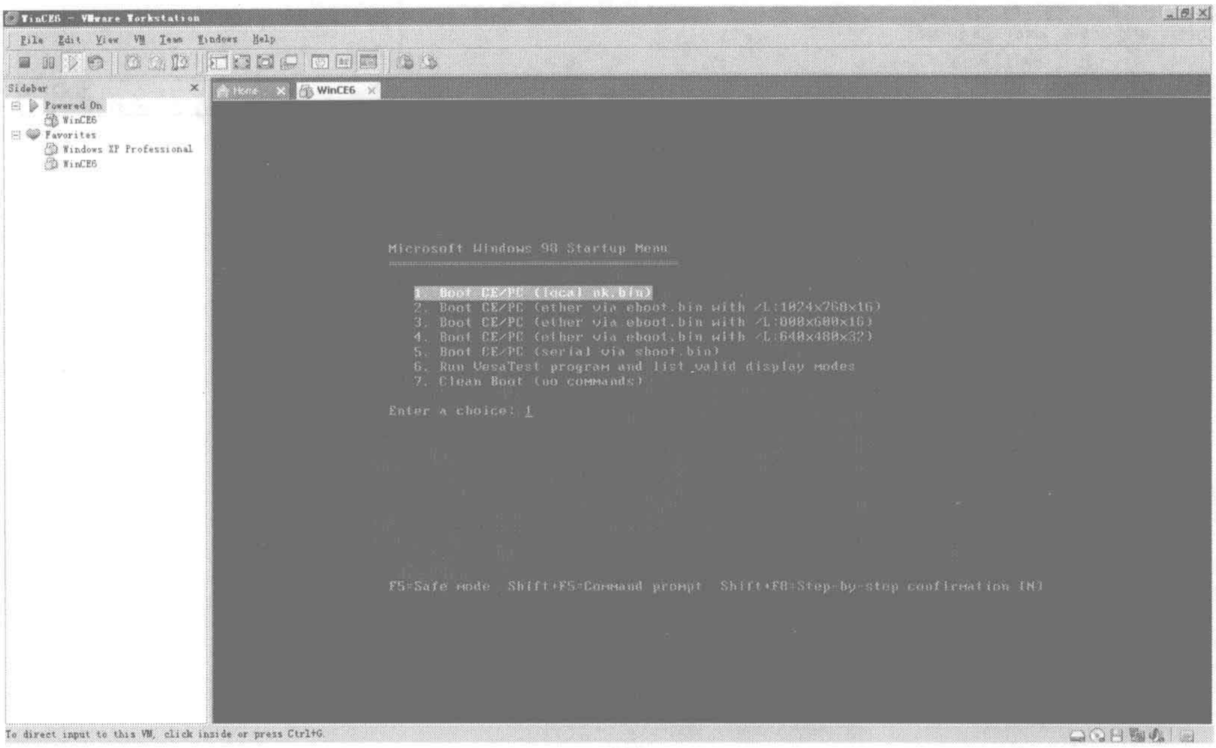


图 3-26

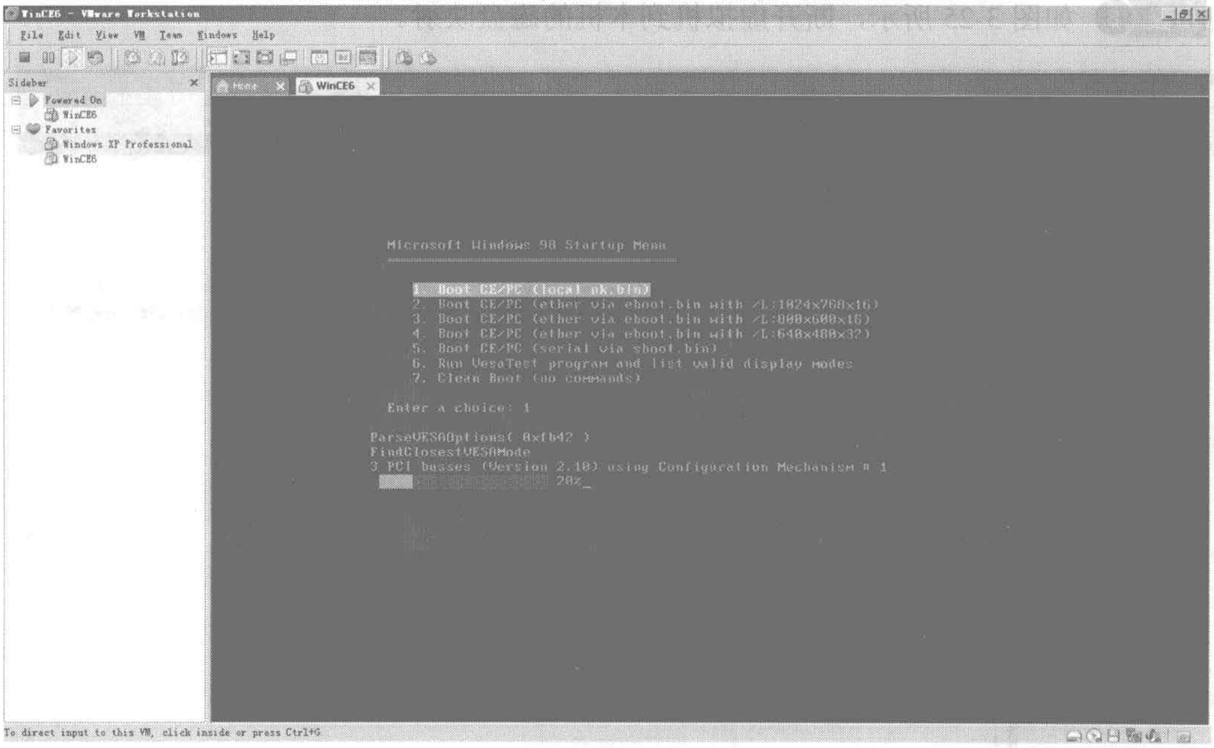


图 3-27

步骤 10 WINCE 启动成功，可以看到虚拟机上的声卡、显卡、硬盘、网卡均被成功驱动，如图 3-28 和图 3-29 所示（USB 也被成功驱动，没有截图）。

图 3-28

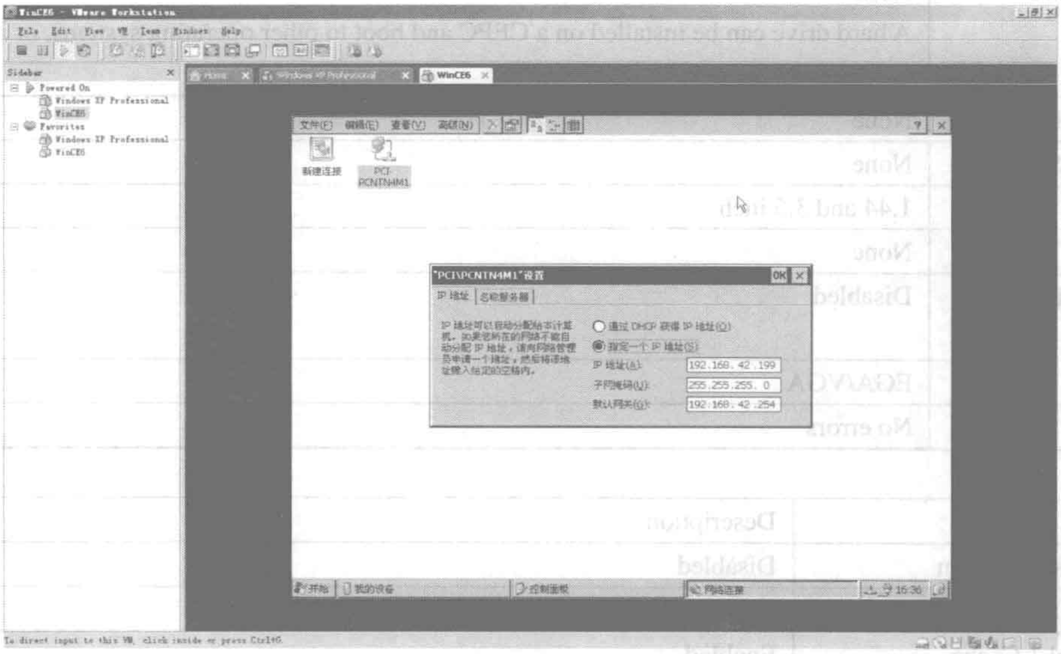


图 3-29

3.2　配置 X86 硬件运行环境

如果需要在真实硬件平台上运行 WINCE6，哪些硬件是 WINCE6 自带的 BSP 支持的呢？实际上，PB6.0 的 CEPC 硬件平台是参考 Asus TUSL2-C Intel 815EP 主板设计的，并且大部分向下兼容。根据参考文档，建议目标硬件平台的 BIOS 参数设置如下表所示，没有列出的项选择默认设置即可。那么，只要是这一系列的兼容主板，WINCE6 应该本身都支持，不需要添加特殊的驱动程序。

以下 BIOS 设置对其他主板有极其重要的参考意义。

BSP	CEPC
Family	x86
CPU	P5, PII, PIII, Celeron, K6x, Athlon
Kernel to use	x86
Manufacturer	Any desktop computer vendor
Part number	Depends on vendor
Floating point unit	Yes
Instruction set	x86
Name in Platform Builder IDE	CEPC: x86
Sample Code Directory	%_WINCEROOT%\Platform\CEPC

Setting	Description
Date	Your local date
Time	Your local time
Primary Master	Auto Supports the ATAPI CD-ROM drive that must be the primary master.
Primary Slave	None A hard drive can be installed on a CEPC and boot to other operating systems, or it can be used to load images.
Secondary Master	None
Secondary Slave	None
Drive A	1.44 and 3.5 inch
Drive B	None
Floppy 3 Mode Support	Disabled
Video	EGA/VGA
Halt On	No errors

Setting	Description
Boot Virus Detection	Disabled
Processor Serial Number	Not applicable
CPU Level 1 Cache	Enabled
CPU Level 2 Cache	Enabled
CPU Level 2 Cache ECC Check	Disabled
BIOS Update	Enabled
Quick Power On Self Test	Enabled
HDD Sequence SCSI/IDE First	IDE
Boot Sequence	A, CDROM, C It is possible to create bootable CD-ROMs and load Nk.bin from the CD-ROM
Boot Up Floppy Seek	Disabled
Floppy Disk Access Control	R/W
IDE HDD Block Mode Sectors	HDD MAX
HDD S.M.A.R.T. Capability	Disabled

PS/2 Mouse Function Control	Auto
OS/2 Onboard Memory > 64M	Disabled
PCI/VGA Palette Snoop	Disabled
Video ROM BIOS Shadow	Enabled
C8000-CBFFF	Disabled
CC000-CFFFF	Disabled
D0000-D3FFF	Disabled
D4000-D7FFF	Disabled
D8000-DBFFF	Disabled
DC000-DFFFF	Disabled
Boot Up NumLock Status	On
Typematic Rate Setting	Disabled
Typematic Rate（Chars/Sec）	6
Typematic Delay （Msec）	250
Security Option	System

Setting	Description
SDRAM Configuration	By SPD
CDRAM CAS Latency	By SPD
SDRAM RAS to CAS Delay	By SPD
SDRAM RAS Prechard Time	By SPD
DRAM Idle Timer	8T
DRAM MA Wait State	Normal
Snoop Ahead	Enabled
Host Bus Fast Data Ready	Disabled
16-bit I/O Recovery Time	1 BUSCLK
8-bit I/O Recovery Time	1 BUSCLK
Graphics Aperture Size	64MB
Video Memory Cache Mode	UC
PCI 2.1 Support	Enabled
Memory Hole at 15M-16M	Disabled
Onboard FDC Controller	Enabled
Onboard FDC Swap A&B	No Swap
Onboard Serial Port 1	3F8H/IRQ4
Onboard Serial Port 2	2F8H/IRQ3
Onboard Parallel Port	378H/IRQ7
Parallel Port Mode	ECP+EPP
ECP DMA Select	3
UART2 Use Infrared	Disabled
Onboard PCI IDE Enable	Primary. This disables the secondary IDE channel.
IDE Ultra DMA Mode	Auto
IDE0 Master PIO/DMA Mode	Auto
IDE0 Slave PIO/DMA Mode	Auto

IDE1 Master PIO/DMA Mode	Auto
IDE1 Slate PIO/DMA Mode	Auto

Setting	Description
Power Management	User Defined
Video Off Option	Suspend->Off
Video Off Method	DPMS Off
HDD Power Down	Disabled
Suspend Mode	Disabled
PWR Button < 4 seconds	Soft Off
PNP OS Installed	No. This is required.
Slot 1 IRQ	If IRQ = 0 and IoBase = 0 in AutoExec.bat on the boot disk, the boot loader searches for the Ethernet Debug Card.
Slot 2 IRQ	Auto
Slot 3 IRQ	Auto
Slot 4 IRQ	Auto
PCI Latency Timer	0 PCI Clock
IRQ 3 Used by ISA	No/ICU
IRQ 4 Used by ISA	No/ICU
IRQ 5 Used by ISA	Yes. This is for the ISA NE2000 NIC. IRQ 5 and base address 340h are used; however, you can use a different IRQ and base address if you modify the call to LoadCEPC.exe on your boot disk.
IRQ 7 Used by ISA	No/ICU
IRQ 9 Used by ISA	No/ICU
IRQ 10 Used by ISA	No/ICU
IRQ 11 Used by ISA	Yes, this IRQ is used by the ISA-based PCMCIA reader.
IRQ 12 Used by ISA	No/ICU
IRQ 14 Used by ISA	No/ICU
IRQ 15 Used by ISA	No/ICU
DMA 1 Used by ISA	No/ICU
DMA 3 Used by ISA	No/ICU
DMA 5 Used by ISA	No/ICU
ISA MEM Block BASE	No/ICU
SYMBIOS SCSI BIOS	Auto
USB IRQ	Enabled
VGA BIOS	PCI/AGP
Sequence	Enabled
PWR Up On Modem Act	Disabled
AC PWR Loss	Disabled

Restart	
Power Up By Keyboard	Disabled
Wake On LAN	Disabled
Automatic Power Up	Disabled

综上所述，WINCE6 的标准 x86 BSP 可兼容许多硬件平台。并不需要特定的嵌入式主板也可以开发 WINCE6 的操作系统与应用程序。本人在学习过程中，大多采用的是研华 PCM3370 主板，直接使用 WINCE6 标准 BSP，各项功能可以全部支持。另外，本人也测试了十多款家用 PC 机主板、CPU、外设等，绝大部分都可以启动 WINCE6，但也有少部分硬盘、网络、鼠标等支持得不是很好。

另外，对于一些特殊硬件，微软也提供了驱动程序。自 WINCE 发布以来，各个版本支持的一部分硬件如下表所示，但这并不完全，比如打印机，实际上所有支持 PCL 打印语言的都可以被正确驱动。当然，有些驱动并没有包含在 WINCE6 当中，但这并不妨碍我们将其移植到 WINCE6 当中来。比如前面提到的 VMWare 虚拟机网卡，AMD 提供的依然是 WINCE2.11 版本的驱动，在 WINCE6 当中它依然可以正常工作。

Type	Company Name	Product Name
1394 Controller	ADS Technologies, Inc.	Pyro 1394DV [PCI]
1394 Controller	D-Link	DFW-500 [PCI]
1394 Controller	Orange Micro	OrangeLink [PCI]
Audio	Creative	AWE64-Compatible 16-bit Audio [PCI]
Audio	Creative	AudioPCI / SoundBlaster 16 [PCI]
Audio	Tvia	CyberPro 5050 [PCI]
Display	Generic Display Adapter	All Vesa 2.0 VGA Compatible Video Cards
Display	3D Labs	Oxygen VX1
Display	3Dfx	Voodoo3 3000
Display	ATI	Rage Expert2000
Display	ATI	Rage XL（Victoria PCI）
Display	ATI	RageExpert98
Display	ATI	RageFury
Display	ATI	RagePro
Display	Creative Labs	TNT2
Display	Creative Labs	TNT2 Ultra
Display	Diamond Multimedia	Diamond Stealth S540
Display	Diamond Multimedia	Diamond Viper V770
Display	Media Q	MQ200
Display	Number9	SR9
Display	Tvia	CyberPro 5050
Input/Game	Microsoft	Gamepad [USB]

Input/Game	Microsoft	Gamepad Pro [USB]
Input/Keyboard	Behavior Tech Computer (BTC)	Keyboard [USB]
Input/Keyboard	Generic	Generic PS/2 keyboard [PS/2]
Input/Keyboard	Generic keyboard	Generic keyboard [USB]
Input/Keyboard	Genius	Genius keyboard [USB]
Input/Keyboard	Justy Japanese	Justy Japanese [USB]
Input/Keyboard	Microsoft	Internet Keyboard Pro [USB]
Input/Keyboard	Microsoft	Natural Keyboard [USB]
Input/Keyboard	Microsoft	Natural Keyboard Pro [USB]
Input/Pointing Drawing	Generic	Generic PS/2 mouse [PS/2]
Input/Pointing Drawing	Microsoft	IntelliMouse [USB]
Modem/Analog	3Comm U.S. Robotics	Courier V-Everything 56K serial modem
Modem/Analog	3Comm U.S. Robotics	Model No: 3CXM756 [PCMCIA]
Modem/Analog	3Comm U.S. Robotics	Sportster 56K serial modem
Modem/Analog	3Comm/Megahertz	XJ4288 [PCMCIA]
Modem/Analog	Socket	Model No: FB BCM 560-LF [CF]
Modem/Analog	Xircom	CreditCard Modem 33.6K [CF]
Modem/Analog	Xircom	Model No: CM-56 [PCMCIA]
Network/BlueTooth	Brainboxes	Bluetooth CF Card [PCMCIA]
Network/BlueTooth	CSR	Microsira USB Dongle [USB]
Network/BlueTooth	Ericsson	Bluetooth UART Dongle [USB] [Serial]
Network/BlueTooth	Socket	Bluetooth CF Card [PCMCIA]
Network/Ethernet	D-link	DFE-530TX+ 10/100 Mb Ethernet NIC Adapter with Wake on LAN [PCI]
Network/Ethernet	Kingston	20T/20BT [ISA] [NE2000]
Network/Ethernet	Kingston	30T/30BT [PCI] [NE2000]
Network/Ethernet	Kingston	KNE100 [PCI]
Network/Ethernet	Linksys	EtherPCI LAN Card II [PCI]
Network/Ethernet	Linksys	Ether16 10BaseT LAN Card [ISA]
Network/Ethernet	Linksys	LNE2000 [ISA] [NE2000]
Network/Ethernet	Netgear	FA-311, FA-312 [PCI] [3.3v]
Network/Ethernet	Socket	LowPower Ethernet CF+ [CF] [NE2000]
Network/Ethernet	Socket	LowPower Ethernet LP-E [CF] [NE2000]
Network/Ethernet	Xircom	CompactCard, Ethernet 10 [CF]
Network/Ethernet	Xircom	Performance Series Credit Card Ethernet Adapter IIps , Model PS-CE2 [PCMCIA]

Network/IrDA	Extended Systems	XTNDAccess Serial [PCMCIA] [IrDA]
Network/IrDA	Hitachi	Hitachi HD64465 FIR [IrDA]
Network/IrDA	NEC	NEC Vr41xx FIR [IrDA]
Network/Wireless	Cisco Systems	340 Series Wireless LAN Adapter [PCMCIA]
Network/Wireless	Cisco Systems	350 Series Wireless LAN Adapter [PCMCIA]
Miscellaneous/Device Sync	Socket Comm	USB Sync Card [CF]
Miscellaneous/Multiport Serial Adapter	Socket Comm	Dual Serial I/O PC Card （Multifunction operation）
Miscellaneous/Multiport Serial Adapter	Socket Comm	Combination I/O PC Card （Single function operation）
Miscellaneous/ USB Function	NetChip	Net2890 USB Interface Controller [PCI]
Miscellaneous/PCMCIA Add-in Adapter	ActionTec	PC700 [ISA]
Miscellaneous/PCMCIA Add-in Adapter	ActionTec	PC750 [PCI]
Miscellaneous/PCMCIA Add-in Adapter	Envoy Data	SBP-D2 PCI SwapBox [PCI]
Miscellaneous/PCMCIA Add-in Adapter	LinkSys	ProConnect Series [ISA]
Miscellaneous/PCMCIA Add-in Adapter	Sandisk	CompactFlash PC Card Adapter [PCMCIA]
Miscellaneous/PCMCIA Add-in Adapter	Simple Technology	Simple Station [ISA]
Miscellaneous/Secure Digital Add-in Adapter	Sandisk	SDSDEV-01 [PCI]
Printer	Canon	BJC-6000 [USB]
Printer	Canon	BJC-85 [USB]
Printer	Epson	Stylus COLOR 740 ESC/P 2
Printer	Hewlett-Packard	Deskjet Model 340C [Parallel]
Printer	Hewlett-Packard	Deskjet Model 400 [Parallel]
Printer	Hewlett-Packard	Deskjet Model 4000 [USB]
Printer	Hewlett-Packard	Deskjet Model 600 [Parallel]

Printer	Hewlett-Packard	Deskjet Model 810C [USB]
Printer	Hewlett-Packard	Laserjet Model 4Si [Parallel]
Printer	Hewlett-Packard	Laserjet Model 5P [Parallel]
Printer	Hewlett-Packard	Laserjet Model 5Si [Parallel]
Printer	Hewlett-Packard	Laserjet Model 6P [Parallel]
Printer	Lexmark	Optra E312L [USB]
Printer	Lexmark	Z51 Color Jetprinter [USB]
Smart Card Reader	Bull	SmarTLP3 Serial Smart Card Reader [Serial]
Smart Card Reader	SCM Microsystems	SCR300 USB Smart Card Reader [USB]
Smart Card Reader	SCM Microsystems	SwapSmart [PCMCIA]
Storage/1394	ADS Tech	Pyro 1394 Drive Kit
Storage/1394	Evergreen	FireLink
Storage/1394	LaCie	PocketDrive 10GB
Storage/1394	VST	FW9530-G
Storage/1394	VST	FW12300-G
Storage/CD-ROM Drive	Hitachi	
Storage/CD-ROM Drive	Matsushita/Panasonic	
Storage/CD-ROM Drive	Toshiba	
Storage/Flash Memory	Intel	Intel 3v and 5v StrataFlash [RFA] [PSM]
Storage/Flash Memory	Intel	Intel 5v StrataFlash [RFA]
Storage/Flash Memory	Intel	Series 2+ StrataFlash Card [PCMCIA]
Storage/Flash Memory	M-Systems	DiskOnChip
Storage/Flash Memory	Hitachi	Flash PC Card [PCMCIA]
Storage/Flash Memory	Kingston	Flash PC Card [PCMCIA]
Storage/Flash Memory	MagicRam	Turbo Flash PC Card [PCMCIA]
Storage/Flash Memory	New Media	Flash CF Card [PCMCIA]
Storage/Flash Memory	SanDisk	Flash PC Card [PCMCIA]
Storage/Flash Memory	SanDisk	CF Series CF Card [CF]
Storage/Flash Memory	Simple Technology	ATA Flash [PCMCIA]
Storage/Flash Memory	Verbatim	Flash PC Card [PCMCIA]
Storage/Flash Memory	M-Systems	Compact Flash Card [CF]
Storage/Flash Memory	M-Systems	FlashLite 100 PC Card [PCMCIA]
Storage/Hard Drive	Callunacard	512MB HD Type III PC Card [PCMCIA]
Storage/Hard Drive	IBM	Microdrive Memory Card [PCMCIA]
Storage/Hard Drive	MagicRam	260MB HD Type III PC Card [PCMCIA]
Storage/IDE Controller	Promise	Ultra66 IDE Controller [PCI]

Storage/IDE Controller	Promise	Ultra100 IDE Controller [PCI]
Storage/IDE Controller	Intel	82371 [PCI]
Storage/IDE Controller	Acer Laboratories Inc	1543 [PCI]
Storage/Removable Drive	IOMega	Peerless [USB]
Storage/Removable Drive	LaCie	PocketDrive 10GB [USB]
Storage/Removable Drive	M-Systems	DiskOnKey [USB]
Storage/SRAM	Simple Technology	SRAM PC Card 6mb [PCMCIA]
USB Hub	Computer Access Technology Corporation	Andromeda USB Hub [USB] [non-powered]
USB Hub	ADS technologies	4 port USB Hub [USB]
USB Hub	Entrega	7 port USB Hub [USB]
USB Hub	Linksys	4 port USB Hub [USB]
USB Hub	NEC	USB Hub [USB]
USB Hub	Northstar Systems	HubStar 4 port Hub [USB]
Video Capture / TV Tuner	Orange Micro	iBOT [1394]
Video Capture / TV Tuner	Panasonic	PV-DV100 [1394]
Video Capture / TV Tuner	Panasonic	PV-DV101D [1394]

3.3 运行时调试操作系统

WINCE6 操作系统的调试，首先是通过 EBOOT.bin 来实现的，EBOOT.bin 通过 tftp 协议从开发机上下载 nk.bin 文件，然后跳转到 nk.bin 的启动地址，然后交由 KITL 负责，实现操作系统在线调试。

需要注意的是，编译的目标 NK 文件必须是调试版，目标硬件的网卡必须有驱动，并且对 KITL 有良好的支持，如图 3-30 所示。

建议目标硬件平台最好有 RTL8139 网卡的支持，因为 EBOOT.bin 默认添加了该网卡的驱动程序。若目标硬件平台是其他网卡，需要修改 EBOOT 的源代码获得支持。

实际上，有了上述各项支持，在 VS2005 平台上，调试 WINCE6 操作系统，就像调试普通应用程序一样。

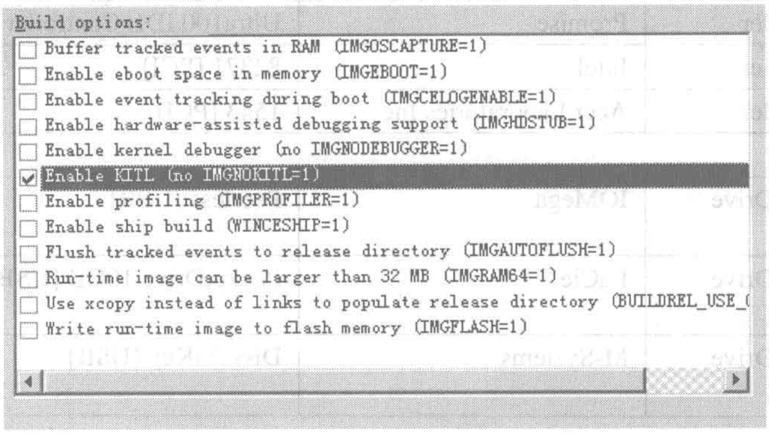


图 3-30

VMWare 虚拟机的网卡由于缺少 EBOOT 和 KITL 的支持，因此不修改源代码的情况下是不能调试操作系统的。

如果不是非常有必要，不建议调试 WINCE 操作系统，绝大多数问题并不需要我们去修改操作系统的内核源码来解决。在 Release 版本状态下足以解决工程应用上的绝大部分问题。

当正确进行了以上设置之后，打开 VS2005 平台，选择“Target”→“Connectivity Options”选项，如图 3-31 所示，在弹出的窗口中的“Download”和“Transport”下拉列表框中选择“Ethernet”选项，设置成网络下载，将“Debugger”设置为“Kdstub”，如图 3-32 所示。然后单击“Download”旁边的“Settings”按钮，在弹出的窗口中等待目标机开机发送“Boot Me”的广播包，如图 3-33 所示。

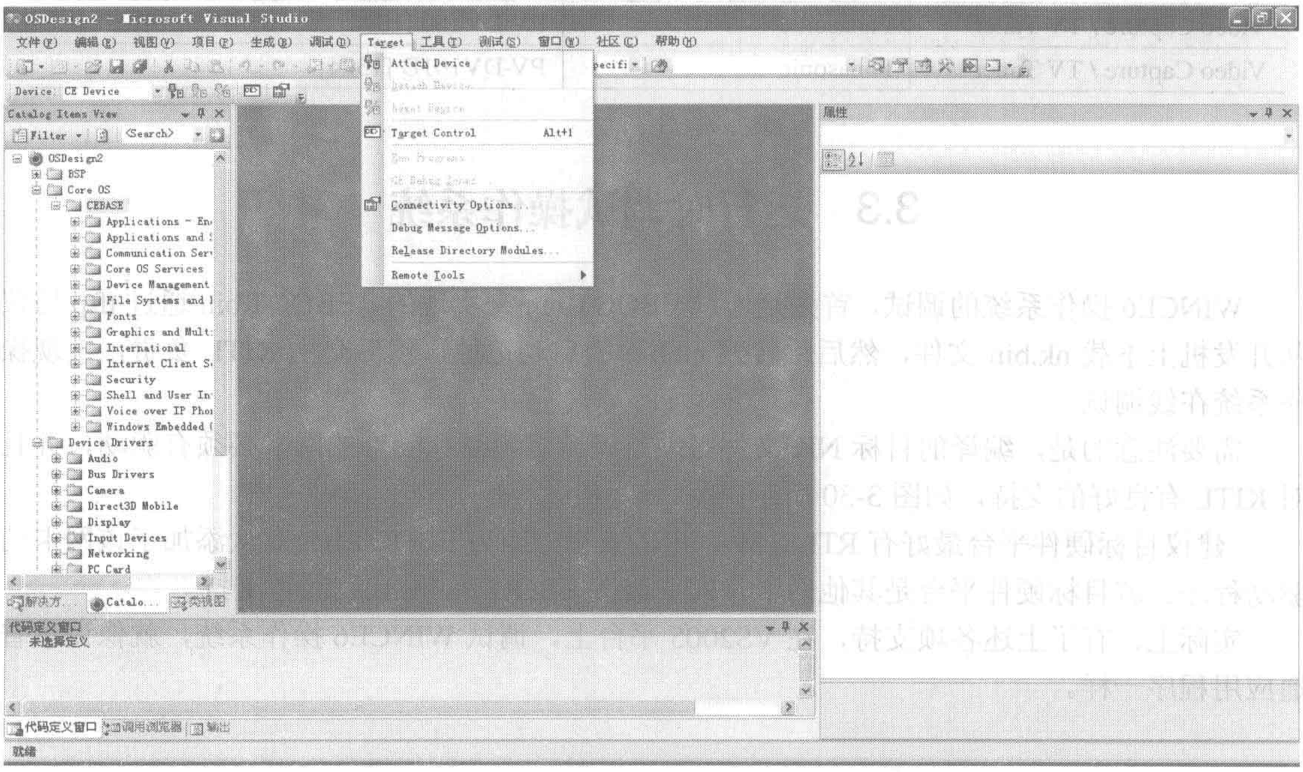


图 3-31

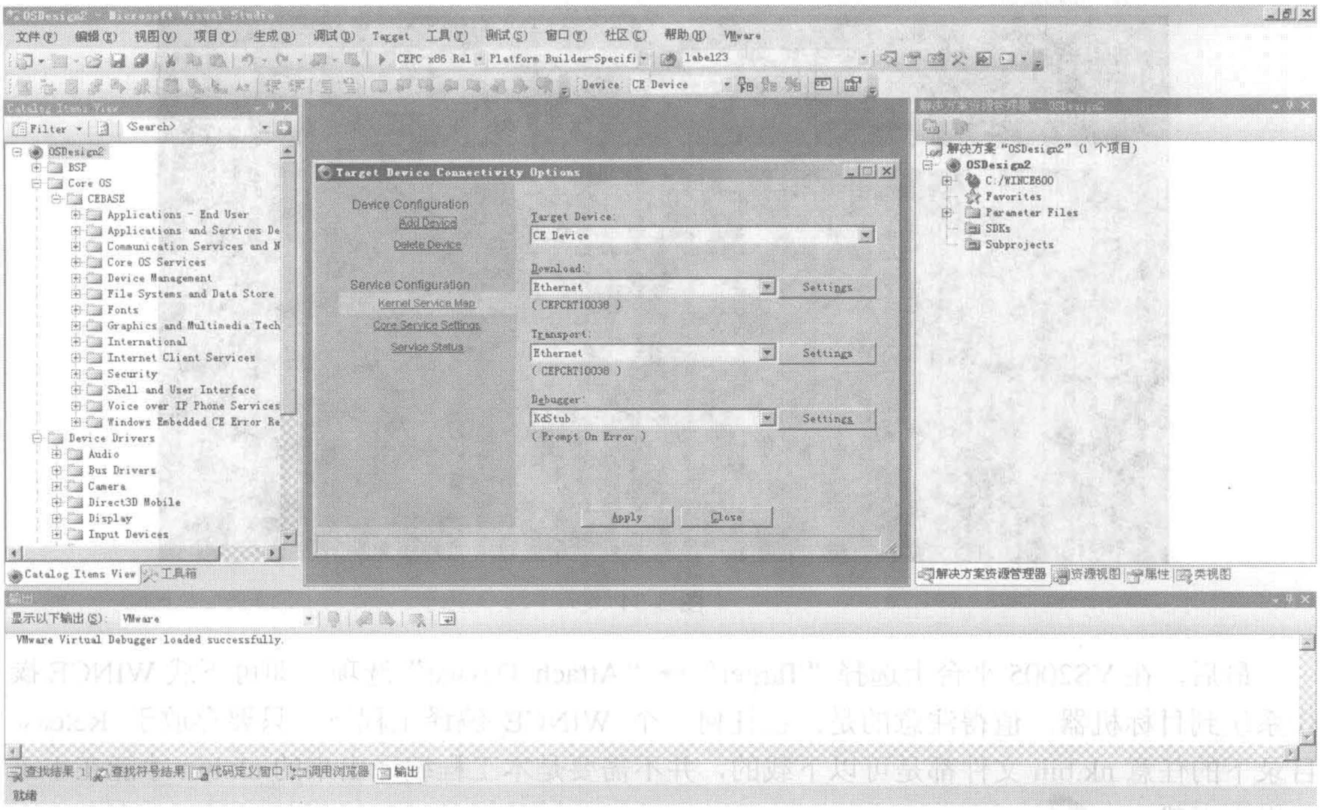


图 3-32

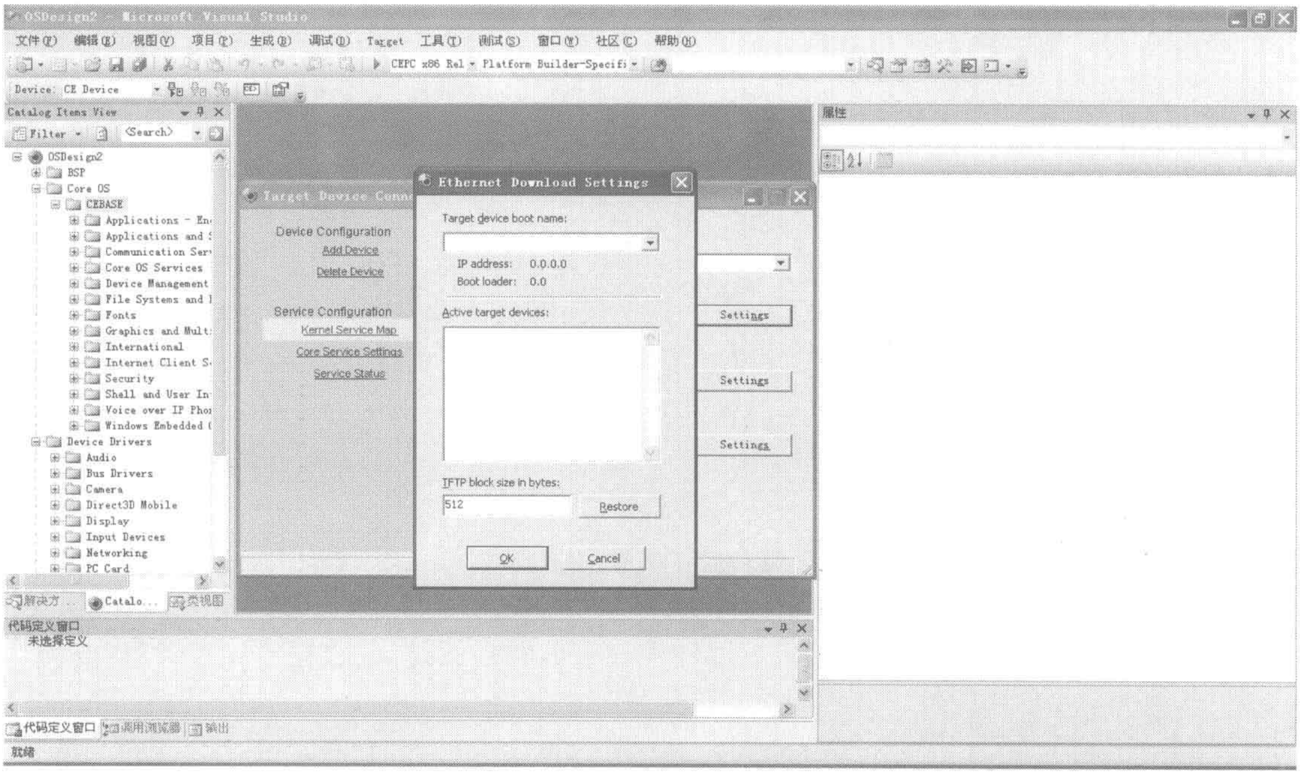


图 3-33

设置好开发平台环境之后，将目标机器开机，在 DOS 启动菜单中选择第三项，利用 EBoot 从网络上下载 nk.bin 文件，如图 3-34 所示。

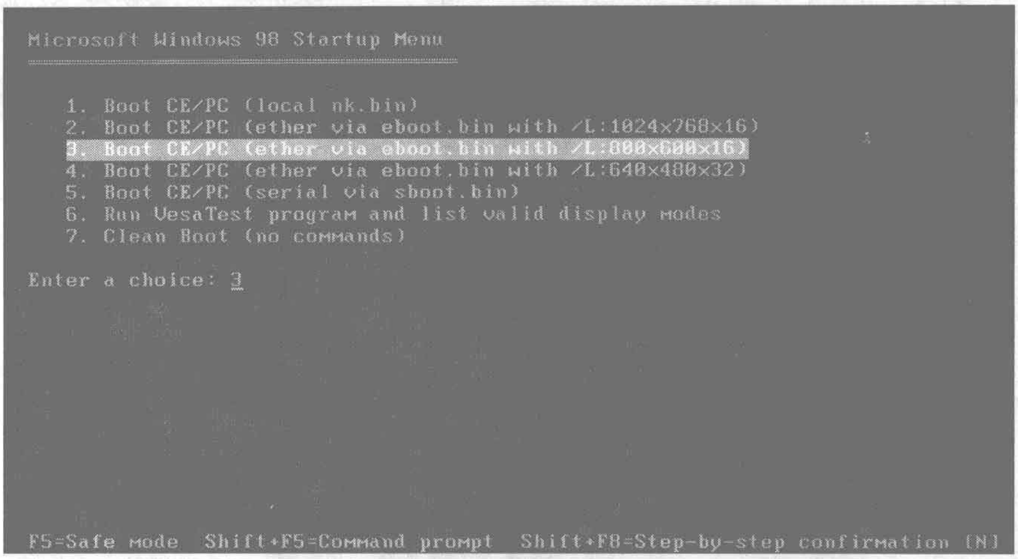


图 3-34

最后，在 VS2005 平台上选择“Target”→“Attach Device”选项，即可下载 WINCE 操作系统到目标机器。值得注意的是，在任何一个 WINCE 编译工程下，只要存放于 Release 目录下的任意 nk.bin 文件都是可以下载的，并不需要是本工程编译出来的 nk.bin 才能在本工程当中下载。

第 4 章　BootLoader 的开发

从本章开始，将正式进入编程阶段。X86 平台上的 BootLoader 的开发并不是必需的，并且 WINCE 上的 BootLoader 开发也不同于其他嵌入式操作系统，不同于其他硬件平台。实际上，不同的硬件平台只需要对微软提供的 Eboot 稍作修改即可实现系统的引导。本章将以实例的方式对 PXA270 平台的 Eboot 开发作详细介绍并提供全部的源代码。由于本章的源代码当中包含了大量注释，足以说明程序的结构与功能，故正文部分未作详细解释。

4.1　BootLoader 介绍

BootLoader 是系统加电运行的第一段软件代码。回忆一下 X86 的体系结构我们可以知道，PC 机中的引导加载程序由 BIOS 和位于硬盘 MBR 中的引导程序一起组成。BIOS 在完成硬件检测和资源分配后，将硬盘 MBR 中的引导程序读到系统的 RAM 中，然后将控制权交给引导程序。引导程序的主要运行任务就是将内核映象从硬盘上读到 RAM 中，然后跳转到内核的入口点去运行，即开始启动操作系统。

因此，在介绍 BootLoader 之前，先简单介绍一下 PC 机上的 BIOS。实际上，X86 体系结构的 BootLoader 完全可以由 BIOS 来替代，当然，也可以用 BootLoader 来替代 BIOS。

BIOS 是英文“Basic Input Output System”的缩略语，直译过来后中文名称就是“基本输入输出系统”。它的全称应该是 ROM－BIOS，意思是只读存储器基本输入输出系统。其实，它是一组固化到计算机主板上一个 ROM 芯片上的程序，它保存着计算机最重要的基本输入输出的程序、系统设置信息、开机上电自检程序和系统启动自举程序。 其主要功能是为计算机提供最底层的、最直接的硬件设置和控制。BIOS 设置程序是储存在 BIOS 芯片中的，只有在开机时才可以进行设置。CMOS 主要用于存储 BIOS 设置程序所设置的参数与数据，而 BIOS 设置程序主要对计算机的基本输入输出系统进行管理和设置，使系统运行在最好状态下，使用 BIOS 设置程序还可以排除系统故障或者诊断系统问题。形象地说，BIOS 应该是连接软件程序与硬件设备的一座“桥梁”，负责解决硬件的即时要求。主板上的 BIOS 芯片或许是主板上唯一贴有标签的芯片，一般它是一块 32 针的双列直插式的集成电路，上面印有“BIOS”字样。586 以前的 BIOS 多为可重写 EPROM 芯片，上面的标签起着保护 BIOS 内容的作用（紫外线照射会使 EPROM 内容丢失），不能随便撕下。586 以后的 ROM BIOS 多采用 EEPROM（电可擦写只读 ROM），通过跳线开关和系统配带的驱动程序盘，可以对 EEPROM 进行重写，方便地实现 BIOS 升级。计算机用户在使用计算机的过程中，都会接触到 BIOS，它在计算机系统中起着非常重要的作用。一块主板性能优越与否，很大程度上取决于主板上的 BIOS 管理功能是否先进。

然而，在其他非 X86 硬件体系结构的嵌入式系统中，通常并没有像 BIOS 那样的固件程序，因此整个系统的加载启动任务就完全由 BootLoader 来完成。也就是说，若 BootLoader 需要足够的复杂程序，完全可以做得像 BIOS 一样。

对于一个嵌入式系统来说，可能有的包括操作系统，有的小型系统也可以只包括应用程序，但是在这之前都需要 BootLoader 为它准备一个正确的环境。通常，BootLoader 是依赖于硬件而实现的，特别是在嵌入式领域，为嵌入式系统建立一个通用的 BootLoader 是很困难的。

一般来说，对于 BootLoader 的功能要求并不是严格定义的，Boot Loader 就是在操作系统内核运行之前运行的一段小程序。通过这段小程序，可以初始化硬件设备、建立内存空间的映射图，从而将系统的软硬件环境带到一个合适的状态，以便为最终调用操作系统内核准备好正确的环境。但不同的场合区别很大。比如，在 PC 的硬件平台上，由于硬件启动根本就不是通过 BootLoader（而是通过 BIOS），所以 BootLoader 就不需要对 CPU 加电后的初始化做任何工作。

通常，Boot Loader 是严重地依赖于硬件而实现的。因此，在各种不同的嵌入式平台上建立一个通用的 Boot Loader 几乎是不可能的。尽管如此，Boot Loader 仍有一些通用的特性可供参考，可以指导用户特定的 Boot Loader 设计与实现。

（1） Boot Loader 所支持的 CPU 和嵌入式板

每种不同的 CPU 体系结构都有不同的 Boot Loader。有些 Boot Loader 也支持多种体系结构的 CPU，比如 U-Boot 就同时支持 ARM 体系结构和 MIPS 体系结构。除了依赖于 CPU 的体系结构外，Boot Loader 实际上也依赖于具体的嵌入式板级设备的配置。这也就是说，对于两块不同的嵌入式板而言，即使它们是基于同一种 CPU 而构建的，要想让运行在一块板子上的 Boot Loader 程序也能运行在另一块板子上，通常都需要修改 Boot Loader 的源程序。

（2） Boot Loader 的位置

系统加电或复位后，所有的 CPU 通常都从某个由 CPU 制造商预先安排的地址上取指令。比如，基于 ARM7TDMI core 的 CPU 在复位时通常都从地址 0x00000000 取它的第一条指令。而基于 CPU 构建的嵌入式系统通常都有某种类型的固态存储设备（比如：ROM、EEPROM 或 FLASH 等）被映射到这个预先安排的地址上。因此在系统加电后，CPU 将首先执行 Boot Loader 程序。

（3） 用来控制 Boot Loader 的设备

主机和目标机之间一般通过串口、USB 口或者网口建立连接，Boot Loader 软件在执行时通常会通过串口 USB 口或者网口来进行 I/O，比如：输出打印信息到串口，从串口读取用户控制字符等。

（4） Boot Loader 的操作模式（Operation Mode）

大多数 Boot Loader 都包含两种不同的操作模式："启动加载模式"和"下载模式"，这种区别仅对于开发人员才有意义。但从最终用户的角度看，Boot Loader 的作用就是用来加载操作系统，而并不存在所谓的启动加载模式与下载工作模式的区别。

启动加载（Boot loading）模式：这种模式也称为"自主" 模式。也即 Boot Loader 从目标机上的某个固态存储设备上将操作系统加载到 RAM 中运行，整个过程并没有用户的介入。这种模式是 Boot Loader 的正常工作模式，因此在嵌入式产品发布的时侯，Boot Loader 显然必须工作在这种模式下。

下载（Downloading）模式：在这种模式下，目标机上的 Boot Loader 将通过串口连接或网络连接等通信手段从主机（Host）下载文件，比如：下载内核映像和根文件系统映像等。从主机下载的文件通常首先被 Boot Loader 保存到目标机的 RAM 中，然后再被 Boot Loader 写到目标机上的 FLASH 类固态存储设备中。Boot Loader 的这种模式通常在第一次安装内核与

根文件系统时被使用；此外，以后的系统更新也会使用 Boot Loader 的这种工作模式。工作于这种模式下的 Boot Loader 通常都会向它的终端用户提供一个简单的命令行接口。

（5） BootLoader 与主机之间进行文件传输所用的通信设备及协议

最常见的情况就是，目标机上的 Boot Loader 通过串口与主机之间进行文件传输，传输协议通常是 xmodem / ymodem / zmodem 协议中的一种。但是，串口传输的速度是有限的，因此通过以太网连接并借助 TFTP 协议来下载文件是个更好的选择。

此外，主机方所用的软件也要考虑。比如，在通过以太网连接和 TFTP 协议来下载文件时，主机方必须有一个软件用来的提供 TFTP 服务

4.2 EBoot 开发例程

对于 X86 的体系结构，微软给出的本地引导方式有两种，DOS+Loadcepc 和 BIOS Boot Loader 两种。网络引导方式有 Eboot，另外还有通过串口下载的 Sboot。

在其他体系结构的硬件平台上，BootLoader 通常有开源解决方案，但一般不包括引导 nk.bin 这一部分。因此，其他体系结构的硬件平台通常是以 BootLoader+Eboot 的方式来启动。事实上，WINCE6 的 Eboot 可以独立完成 WINCE 启动的全过程，并不需要其他的 BootLoader 来支持。只是 Eboot 功能比较单一，可供发挥的余地较小，也不支持多系统引导等功能，若只是用来引导 WINCE，Eboot 已经足够。但针对特定的板卡，还是需要修改源码才能让 Eboot 正常工作。

在 WINCE6 平台中，附带有包括 X86、ARM、SH、MIPS、XScale 等各个硬件平台的 Eboot 的大部分源码。当然，部分核心源码仍需要自己完成。

下面，给出一个基于 PXA270 平台的 Eboot 范例。在本例当中，主要实现了两大功能。第一个是从网卡下载 nk.bin 文件并启动 WINCE 系统，第二个是从 SD 卡上查找 nk.bin 文件并启动 WINCE 系统。有启动界面和加载进度条。

本例文件结构如图 4-1 所示。

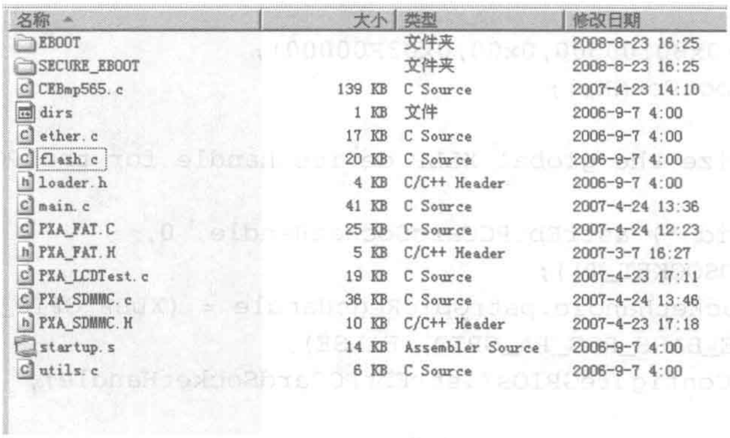


图 4-1

WINCE6 中自带的 Eboot 文件结构如图 4-2 所示。

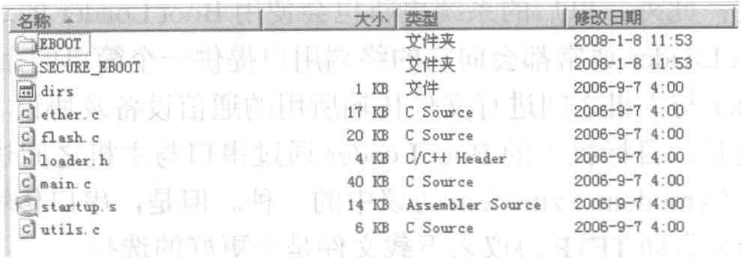


图 4-2

实际上，前文所述 BootLoader 开发中大部分工作都已经完成，相对而言，只需要稍作修改即可完成 Eboot 的开发。

下面给出修改部分的源代码说明。

主要修改有以下几处：Main.c 文件当中，修改了函数 OEMPlatformInit：添加了 SD 卡文件系统读写；添加了 LCD 显示驱动；添加了启动画面。

修改后的 OEMPlatformInit 函数源代码如下所示。

```
    BOOL OEMPlatformInit(void)
    {
        UINT32 AutoBootDelay = 0;
        BOOLEAN bXIPMode      = TRUE;
        BOOLEAN bCFGChanged   = FALSE;
        SYSTEMTIME SysTime;
        UINT32 StartTime, CurrTime, PrevTime;
        UINT32 Selection;
        UINT32 BootDevice;
        int i;
        UCHAR  rndis_mac[6] = {0,0,0,0,0,0};
        KITLOutputDebugString("Microsoft Windows CE Ethernet Bootloader %d.%d for
the Intel MainstoneIII Development Platform Built %s\r\n", \
                            EBOOT_VERSION_MAJOR, EBOOT_VERSION_MINOR,
__DATE__);
    //清除内存信息
    memset((void*)0x80100000,0x00,0x03F00000);
        PXA_InitEBootForNK();

        // Initialize the global XLLP device handle for the PC Card interface.
        //
        memset((void *) &strEbtPCCardSocketHandle, 0,
sizeof(XLLP_PCCARDSOCKET_T));
    strEbtPCCardSocketHandle.pstrGpioRegsHandle = (XLLP_GPIO_T *)
OALPAtoVA(BULVERDE_BASE_REG_PA_GPIO, FALSE);
        XllpPCCardConfigureGPIOs(&strEbtPCCardSocketHandle);

        // Get the Ehternet MAC address to use as RNDIS MAC address
        // Note: This only applies to Mainstone platform because it has Ehternet
controller on it.
        //
        if (!g_EbootCFG.RNDISMac[0] && !g_EbootCFG.RNDISMac[1]
&& !g_EbootCFG.RNDISMac[2])
```

```
    {
        if (LAN91CInit((UINT8 *) (((UINT32)
OALPAtoVA(MAINSTONEII_BASE_REG_PA_SMSC_ETHERNET, FALSE)) + 0x300), 0,
g_EbootCFG.RNDISMac))
        {
            bCFGChanged = TRUE;
        }
    }

    // Read the current wall-clock time.
    // NOTE: At reset, the RTC value is set to 1/1/1980 at 0:0:0.
    //
    OEMGetRealTime(&SysTime);

    // User menu code...
    //
    AutoBootDelay = g_EbootCFG.delay;
    if (g_EbootCFG.autoDownloadImage)
    {
        g_DownloadImage = TRUE;
        KITLOutputDebugString ( "\r\nPress [ENTER] to download now or [SPACE]
to cancel.\r\n");
        KITLOutputDebugString ( "\r\nInitiating image download in %d seconds.
", AutoBootDelay--);
    }
    else
    {
        g_DownloadImage = FALSE;
        KITLOutputDebugString ( "\r\nPress [ENTER] to launch image stored in
flash or [SPACE] to cancel.\r\n");
        KITLOutputDebugString ( "\r\nInitiating image launch in %d seconds.
", AutoBootDelay--);
    }

    // Get a snapshot of the RTC seconds count.
    //
    StartTime     = OEMKitlGetSecs();
    PrevTime      = StartTime;
    CurrTime      = StartTime;
    Selection     = 0;
  {
    DWORD   startTime = OALGetTickCount();
    while ((OALGetTickCount() - startTime) < 2)
    {
        Selection = OEMReadDebugByte();
    }
  }

    // Allow the user an amount of time to halt the auto boot/download process.
    // Count down to 0 before proceeding with default operation.
    //
  /*
```

```
while ((CurrTime - StartTime) < g_EbootCFG.delay)
{
    UINT8 k=0;
    UINT8 j;
    UINT8 x,y,z;

    Selection = OEMReadDebugByte();
    if ((Selection == 0x20) || (Selection == 0x0d))
    {
        break;
    }
    CurrTime = OEMKitlGetSecs();
    if (CurrTime > PrevTime)
    {
        PrevTime = CurrTime;
        if (AutoBootDelay < 9)
            k = 11;
        else if (AutoBootDelay < 99)
            k = 12;
        else if (AutoBootDelay < 999)
            k = 13;

        for (j = 0; j < k; j++)
        {
            OEMWriteDebugByte((BYTE)0x08); // print back space
        }
        x = AutoBootDelay / 100;
        y = (AutoBootDelay % 100) / 10;
        z = ((AutoBootDelay % 100) % 10);
        OEMWriteDebugLED(0, ((x << 8) | (y << 4) | (z)));
        KITLOutputDebugString ( "%d seconds. ", AutoBootDelay--);
    }

}
*/

switch (Selection)
{
case 0x00: // fall through if nothing typed
case 0x0d: // user canceled wait
    {
        if (g_EbootCFG.autoDownloadImage)
        {
            KITLOutputDebugString ( "\r\nStarting auto download ...
\r\n");
        }
        else
        {
            KITLOutputDebugString ( "\r\nLaunching flash image ...
\r\n");
        }
        break;
```

```
            }
        case 0x20:
            {
                Selection = 0;
                while (1)
                {
                    // Get the Ehternet MAC address to use as RNDIS MAC address
                    // Note: This only applies to Mainstone platform because it
has Ehternet controller on it.
                    //
                    if (!g_EbootCFG.RNDISMac[0] && !g_EbootCFG.RNDISMac[1]
&& !g_EbootCFG.RNDISMac[2])
                    {
                        if (LAN91CInit((UINT8 *) (((UINT32)
OALPAtoVA(MAINSTONEII_BASE_REG_PA_SMSC_ETHERNET, FALSE)) + 0x300), 0,
g_EbootCFG.RNDISMac))
                        {
                            bCFGChanged = TRUE;
                        }
                    }

                    // Show menu
                    KITLOutputDebugString ( "\r\n\r\nEBoot Loader
Configuration:\r\n\r\n");

                    KITLOutputDebugString ( "0) IP address:
%s\r\n",inet_ntoa(g_EbootCFG.IP));
                    KITLOutputDebugString ( "1) Subnet mask: %s\r\n",
inet_ntoa(g_EbootCFG.subnetMask));
                    KITLOutputDebugString ( "2) Boot delay: %d seconds\r\n",
g_EbootCFG.delay);
                    KITLOutputDebugString ( "3) DHCP: %s\r\n",
(g_EbootCFG.DHCPEnable == TRUE?"(Enabled)":"(Disabled)"));
                    KITLOutputDebugString ( "4) Reset to factory default
configuration\r\n");
                    KITLOutputDebugString ( "5) RNDIS MAC address:
%x-%x-%x-%x-%x-%x\r\n",(UCHAR) g_EbootCFG.RNDISMac[0], (UCHAR)
(g_EbootCFG.RNDISMac[0]>>8), (UCHAR)g_EbootCFG.RNDISMac[1],

(UCHAR)(g_EbootCFG.RNDISMac[1]>>8), (UCHAR)g_EbootCFG.RNDISMac[2],
(UCHAR)(g_EbootCFG.RNDISMac[2]>>8));
                    KITLOutputDebugString ( "6) %s image at startup\r\n",
g_EbootCFG.autoDownloadImage?"Download new":"Launch existing flash resident");
                    KITLOutputDebugString ( "7) Boot device order: ");
                    {
                        if(g_EbootCFG.bootDeviceOrder >= NUM_BOOT_ORDERS)
                        g_EbootCFG.bootDeviceOrder = 0;

                        KITLOutputDebugString ( "%s\r\n",
BootOrderString[g_EbootCFG.bootDeviceOrder]);
                    }
                    KITLOutputDebugString ( "8) Debug serial port: ");
```

```
                {
                    switch (g_EbootCFG.dwDbgSerPhysAddr)
                    {
                        case BULVERDE_BASE_REG_PA_FFUART:
                        KITLOutputDebugString ( "FFUART\r\n");
                        break;

                        case BULVERDE_BASE_REG_PA_BTUART:
                        default:
                        KITLOutputDebugString ( "BTUART\r\n");
                        break;
                    }
                }
                KITLOutputDebugString ( "D) Download image now\r\n");
                KITLOutputDebugString ( "L) Launch existing flash resident
image now\r\n");
                KITLOutputDebugString ( "M) Launch Image from SDMMC card now
\r\n");
                KITLOutputDebugString ( "R) Refresh LCD or VGA  now\r\n");
                KITLOutputDebugString ( "\r\n\r\nEnter your selection: ");
                Selection = 0;
                while (! ( ( (Selection >= '0') && (Selection <= '8') ) ||
                        ( (Selection == 'D') || (Selection == 'd') ) ||
                        ( (Selection == 'M') || (Selection == 'm') ) ||
                        ( (Selection == 'R') || (Selection == 'r') ) ||
                        ( (Selection == 'L') || (Selection == 'l') ) ))
                {
                    Selection = OEMReadDebugByte();
                }
                KITLOutputDebugString ( "%c\r\n", Selection);
                switch (Selection)
                {
                case 'M':
                case 'm':
                    TestSDMMC();
                    break;
                case 'R':
                case 'r':
                    PXA_TestLCD();
                    break;
                case '0':
                    SetIP(&g_EbootCFG);
                    bCFGChanged=TRUE;
                    break;
                case '1':
                    SetMask(&g_EbootCFG);
                    bCFGChanged=TRUE;
                    break;
                case '2':
                    SetDelay(&g_EbootCFG);
                    bCFGChanged=TRUE;
                    break;
```

```
                case '3':
                    if (g_EbootCFG.DHCPEnable == TRUE)
                        g_EbootCFG.DHCPEnable = FALSE;
                    else
                        g_EbootCFG.DHCPEnable = TRUE;
                    bCFGChanged=TRUE;
                    break;
                case '4':
                    ResetDefaultEBootCFG(&g_EbootCFG);
                    bCFGChanged=TRUE;
                    break;
                case '5':
                    SetMAC(&rndis_mac[0]);
                    // Save the user enter RNDIS MAC address
                    g_EbootCFG.RNDISMac[0] = (rndis_mac[1] << 8) |
rndis_mac[0];
                    g_EbootCFG.RNDISMac[1] = (rndis_mac[3] << 8) |
rndis_mac[2];
                    g_EbootCFG.RNDISMac[2] = (rndis_mac[5] << 8) |
rndis_mac[4];
                    bCFGChanged = TRUE;
                    break;
                case '6':
                    if (g_EbootCFG.autoDownloadImage == TRUE)
                        g_EbootCFG.autoDownloadImage = FALSE;
                    else
                        g_EbootCFG.autoDownloadImage = TRUE;
                    bCFGChanged=TRUE;
                    break;
                case '7':
                    g_EbootCFG.bootDeviceOrder++;
                    if (g_EbootCFG.bootDeviceOrder >= NUM_BOOT_ORDERS)
                        g_EbootCFG.bootDeviceOrder = 0;
                    bCFGChanged=TRUE;
                    break;
                case '8':
                    if ((g_EbootCFG.dwDbgSerPhysAddr == 0) ||
(g_EbootCFG.dwDbgSerPhysAddr == BULVERDE_BASE_REG_PA_BTUART))
                    {
                        g_EbootCFG.dwDbgSerPhysAddr =
BULVERDE_BASE_REG_PA_FFUART;
                    }
                    else
                    {
                        g_EbootCFG.dwDbgSerPhysAddr =
BULVERDE_BASE_REG_PA_BTUART;
                    }
                    bCFGChanged=TRUE;
                    break;
                case 'D':
                case 'd':
                    if (bCFGChanged == TRUE)
```

```
                    {
                        StoreEBootCFG(&g_EbootCFG);
                    }
                    g_DownloadImage = TRUE;
                    goto CONTINUE;
                case 'L':
                case 'l':
                    if (bCFGChanged == TRUE)
                    {
                        StoreEBootCFG(&g_EbootCFG);
                    }
                    g_DownloadImage = FALSE;
                    goto CONTINUE;
                default:
                    break;
            }
        }
    }

    }
  ////////////////////
    PXA_EbootReadSDNK();
  ////////////////////
  CONTINUE:

    // Provide the chosen debug serial port address to the OS image for debug
output messages.
    //
    g_pBSPArgs->dbgSerPhysAddr = g_EbootCFG.dwDbgSerPhysAddr;

    BootDevice = -1;

    // Locate and initialize an Ethernet controller.
    //
    for(i = 0; (i < NUM_BOOT_DEVICES) && (BootDevice == -1) ; i++)
    {
        // Try each of the boot devices in the specified order (until one
succeeds or they all fail).
        //
        BootDevice = InitSpecifiedEthDevice(&g_pBSPArgs->kitl,
BootOrder[g_EbootCFG.bootDeviceOrder][i]);
    }

    if( BootDevice == SDBG_USB_SERIAL )
    {
        // Indicate to OEMPreDownload that serial will be used
        g_SerialUSBDownload = TRUE;
        if(!USBSerialInit())
        {
            KITLOutputDebugString("ERROR: Failed to detect and initialize USB
Function controller.\r\n");
```

```
                return(FALSE);
            }
        }

        // No device was found ...
        //
        if (BootDevice == -1)
        {
            KITLOutputDebugString("ERROR: Failed to detect and initialize
Ethernet controller.\r\n");
            return(FALSE);
        }
        else
        {
            // Make sure MAC address has been programmed.
            //
            if (!g_pBSPArgs->kitl.mac[0] && !g_pBSPArgs->kitl.mac[1]
&& !g_pBSPArgs->kitl.mac[2])
            {
                KITLOutputDebugString("ERROR: Invalid Ethernet address read from
Ethernet controller.\n");
                return(FALSE);
            }
            else
            {
                 KITLOutputDebugString("INFO: MAC address:
%x-%x-%x-%x-%x-%x\r\n",
                        g_pBSPArgs->kitl.mac[0] & 0x00FF, g_pBSPArgs->kitl.mac[0]
>> 8,
                        g_pBSPArgs->kitl.mac[1] & 0x00FF, g_pBSPArgs->kitl.mac[1]
>> 8,
                        g_pBSPArgs->kitl.mac[2] & 0x00FF, g_pBSPArgs->kitl.mac[2]
>> 8);
            }
        }

        return(TRUE);

    }
```

FAT 文件系统定义的相关结构如下所示，对应文件名为 PXA_FAT.H。

```
//-------------------------------------
//描述: PXA270 访问 FAT 文件系统 程序
//日期: 2007-02-27
//修改记录:
//-------------------------------------
#ifndef PXA_270_FAT_2007_02_27_TAYO_CQ__INCLUDE__
#define PXA_270_FAT_2007_02_27_TAYO_CQ__INCLUDE__
////////////////////////////////////////////////////////
#include <windows.h>
```

```
#pragma pack(1)
typedef struct HIDSEC16_TAG
{
 USHORT NumHiddenSectL;
 USHORT NumHiddenSectH;
}HIDSEC16;//保留扇区
#pragma pack(1)
typedef struct TOTSEC16_TAG
{
    USHORT TotalSectorsL;
    USHORT TotalSectorsH;
}TOTSEC16;//总扇区

#pragma pack(1)
typedef struct  FAT16EX_TAG
{

    UCHAR DriveId;//36
    UCHAR  TempVal;
    UCHAR  ExtRecordSig;
    USHORT VolSerNumL;
    USHORT VolSerNumH;
    UCHAR  VolLabel[11];
    UCHAR  TypeFAT[8];
}FAT16EX;//FAT16扩展 offset = 36
#pragma pack(1)
typedef struct  FAT32EX_TAG
{

    DWORD    FATSz32;     //36-4每FAT扇区数(Sectors Per FAT)(只被FAT32使用)该分区
每个FAT所占的扇区数
    WORD ExtFlags;    //40-2
    WORD FSVer;       //42-2
    DWORD    RootClus;   //44-4根目录簇号(Root Cluster Number)(只供FAT32使用) 根
目录第一簇的簇号。
    WORD FSInfo;      //48-2;
    WORD BkBootSec;  //50-2
    UCHAR     Reserved[12];//52-12
 UCHAR    DrvNum;      //64-1
 UCHAR    Reserved1;  //65-1
 UCHAR    BootSig;     //66-1
 UCHAR    VolID[4];    //67-4
 UCHAR    VolLab[11]; //71-11
 UCHAR    FilSysType[8];//82-8
}FAT32EX;//FAT32扩展 offset = 36


#pragma pack(1)
```

```
typedef struct BIOSPB_TAG
{
 BYTE   JMP[3];//跳转指令
    UCHAR  VersionId[8];
    USHORT BytesPerSect;
    UCHAR  SectsPerClust;//每簇扇区数。文件按照簇分配的
    USHORT RsvdSects;//保留扇区数
    UCHAR  NumFATs;//FAT数(Number of FAT) 该分区上FAT的副本数。本字段的值一般为2
    USHORT NumRootEntries;//BPB_RootEntCnt 根目录项数(Root Entries
    USHORT SectsPerPart;//BPB_TotSec16 小扇区数(Small Sector) 该分区上的扇区数
    UCHAR  MediaDesc;//BPB_Media
    USHORT SectsPerFAT;//BPB_FATSz16 每FAT扇区数(Sectors Per FAT)
    USHORT SectsPerTrack;//每道扇区数(Sectors Per Trark)
    USHORT NumHeads;//26 磁头数(Number of Head) 本字段包含使用INT 13h的磁盘的"磁
头数"几何结构值。例如，在一张1.44MB 3.5英寸的软盘上，本字段的值为 2
 union HIDSEC{
    HIDSEC16    HidSec16;
    DWORD       HidSec32;
 } HiddSec;//28 offset size =4 隐藏扇区数(Hidden Sector) 该分区上引导扇区之前的扇
区数
 union TOTSEC{
    TOTSEC16    TotalSec16;
    DWORD       TotalSec32;
 }TotalSec;//32 offset size =4
 union FATEX
 {
    FAT16EX     Fat16Ex;
    FAT32EX     Fat32Ex;
 }FatEx;//36

 } BIOSPB, *PBIOSPB;//BIOSPB分区表;


typedef enum
{
 PXA_FAT_TYPE_NULL=0x00,
 PXA_FAT_TYPE_12 = 0x01,
 PXA_FAT_TYPE_16 = 0x02,
 PXA_FAT_TYPE_32 = 0x03,
}PXA_FAT_TYPE;


#pragma pack(1)
typedef struct PXA_DIRENTRY_TAG
{
 UCHAR    FileName[8];//0-8文件名
 UCHAR    ExtName[3];//8-3扩展名
 UCHAR    FileAttr;//B-1属性
 UCHAR    SysReserved;//系统保留
 UCHAR    MSecond;//创建时间的10毫秒位
 WORD     FileTime;//创建时间
```

```
    WORD     FileDate;//创建日期
    WORD     LastAccessDate;//最后访问日期
    WORD     CulsterHigh;//文件起始簇号的高16位
    WORD     ModiTime;//最近修改时间
    WORD     ModiDate;//最近修改日期
    WORD     FirstClust;//CulsterLow;//文件起始簇号的低16位
    DWORD    FileSize;//文件长度
}PXA_DIRENTRY, *PPXA_DIRENTRY;//文件目录结构


///----------------------------------------
///@brief 1. 初始化FAT系统
///@param pSectorBuffer = PBIOSPB 数据缓冲区，作为系统初始化参数
///@param nReSectors = DPT 保留的扇区数 一般为0x3F。
///----------------------------------------
BOOL PXA_InitFAT(BYTE* pSectorBuffer,WORD nReSectors);

///----------------------------------------
///@brief 2. 打开文件
///@param pFileName = 文件名
///@return 返回实际的文件长度
///----------------------------------------
ULONG PXA_OpenFile(PCHAR pFileName);

///----------------------------------------
///@brief 3. 关闭文件
///----------------------------------------
void PXA_CloseFile(void);

///----------------------------------------
///@brief 4. 读文件
///@param pAddress = 缓冲区地址
///@param Length = 长度
///--------------------------------------------
BOOL PXA_ReadFile(PUCHAR pAddress, ULONG Length);

///------------------------------------------------------------
/// Load a BIN file into memory and jump to the start address.
///@brief 5. 装入BIN文件
///@param pFileName = BIN文件名称
///@param pImageLoc = 返回的BIN文件入口地址
///------------------------------------------------------------
BOOL PXA_LoadBINFile(PCHAR pFileName, PULONG pImageLoc);

//================================================
//相关宏定义信息
#define TO_UPPER(a)      (a >= 0x61 ? a - 0x20 : a) //转换成大写字母
```

```
#endif//PXA_270_FAT_2007_02_27_TAYO_CQ__INCLUDE__
```

FAT 文件系统的实现代码如下所示，对应文件名为 PXA_FAT.C。

```
//----------------------------------------
//描述：PXA270 访问 FAT 文件系统 程序
//日期：2007-02-27
//修改记录：
//----------------------------------------
//#include <windows.h>
#include "PXA_FAT.H"
#define MIN(a, b)         ( (a) < (b) ? (a) : (b) )

#define READ_BUFFER_LENGTH_MAX   0x8000 //一次最多读取的数据个数=32K
//#define READ_BUFFER_START        0x8000    // 开始读的缓冲区地址

#define SECTOR_SIZE              512     // sector size (in bytes).

//+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
//extern void KITLOutputDebugString( const unsigned char *sz, ...);
extern void PXA_WaitPushKey(char cKey );
extern void PXA_ShowGuage(WORD nPercent);//显示进度
//extern void ShowBuffer(BYTE* pBlock);


///@brief 1.显示调试信息
//#define PXA_ShowDebugMessage KITLOutputDebugString
void PXA_ShowDebugMessage( const unsigned char *sz, ...)
{
}

//////////////////////////////////////////////////////////////


///---------------------------------------------------------
///@brief 2. 初始化读参数
///---------------------------------------------------------
static BOOL InitReadBuffer(ULONG StartAddress, ULONG Length);

///---------------------------------------------------------
///@brief 3. 读指定扇区的数据
///@param dwSectorIndex = 扇区开始序号
///@param nSecCount = 扇区个数
///@param pBlockBuffer = 块缓冲区地址
///---------------------------------------------------------
extern BOOL PXA_ReadSectors(DWORD dwSectorIndex,WORD nSecCount ,BYTE*
pBlockBuffer);
```

```
/////////////////////////////////////////////////////////////////
// This data supplements the BPB information - used to save time.
struct
{
 PXA_FAT_TYPE    FATType;            // FAT filesystem type (12, 16, or 32).
    ULONG           FATSector;             // Sector start of FAT (first)
    ULONG           RootDirSector;         // Sector start of root directory
    ULONG           DataStartSector;       // Sector start of data area
} g_FATParms;//FAT参数定义

//
// Structure used to keep track of file properties.
//
typedef struct
{
    ULONG FileSize;//文件长度
 ULONG FirstCluster;//开始文件簇
 ULONG CurCluster;//当前文件簇
 ULONG NumContigClusters;
    PUCHAR pCurReadBuffAddr;
 ULONG NumReadBuffBytes;
} FILEINFO, *PFILEINFO;//文件信息定义

//
// FAT cluster types.
//
typedef enum
{
 EOF_CLUSTER = 1,
 DATA_CLUSTER,
 BAD_CLUSTER,
 RSVD_CLUSTER
} CLUSTTYPE, *PCLUSTTYPE;//文件簇类型


//////////////////////////////////////////
//
// BIN image and record headers
//
#pragma pack(1)
typedef struct            // Image header (one per BIN image)
{
    CHAR SyncBytes[7];
    ULONG ImageAddr;
    ULONG ImageLen;
} IMAGEHDR, *PIMAGEHDR;

#pragma pack(1)
typedef struct            // Record header (one per section in image)
{
    ULONG RecordAddr;
```

```
    ULONG RecordLen;
    ULONG RecordChksum;
} RECORDHDR, *PRECORDHDR;



PBIOSPB         pBPB = NULL;//PBIOSPB指针
PUCHAR          g_pReadBuffStart = 0;//读出数据后存放的地址 在InitReadBuffer()
中初始化
ULONG           g_ReadBuffLenInClusters = 0;//按文件簇为单位的数据缓冲区长度

FILEINFO             g_FileInfo;
static  UCHAR             g_FATCACHE[SECTOR_SIZE]={0};//SECTOR_CACHE_START
FAT分配表缓冲区
static  UCHAR             g_FileReadBuffer[READ_BUFFER_LENGTH_MAX]={0};//文件
读写缓冲区



void ShowBuffer(BYTE* pBlock)
{

 int i;
 return ;
 for( i = 0x00 ; i < 512 ; i++)
 {
     if( i%16==0x00)
     {
         PXA_ShowDebugMessage("\r\n");
     }
     if( i == 256)
     {
 //          WaitSpace();
     }
     PXA_ShowDebugMessage("%x ",pBlock[i]);
 }
 //   WaitSpace();
 PXA_ShowDebugMessage("\r\n");
 }


///--------------------------------------
///@brief 1. 初始化FAT系统
///@param pSectorBuffer = PBIOSPB 数据缓冲区，作为系统初始化参数
///@param nReSectors = DPT 保留的扇区数 一般为0x3F。
///--------------------------------------
BOOL PXA_InitFAT(BYTE* pSectorBuffer,WORD nReSectors)
```

```
{
ULONG   ReadBufferLen = 0;
UCHAR   FATType[9];
WORD    RootDirSectors = 0;

// BIOS parameter block 作为系统初始化的标志缓冲区.
    pBPB = (PBIOSPB) pSectorBuffer;

// Determine FAT filesystem type.
memcpy(FATType, pBPB->FatEx.Fat16Ex.TypeFAT, 8);
FATType[8] = '\0';

if (!memcmp(FATType, "FAT12", 5))
     g_FATParms.FATType              = PXA_FAT_TYPE_12;
else if (!memcmp(FATType, "FAT16", 5))
     g_FATParms.FATType              = PXA_FAT_TYPE_16;
else if (!memcmp(FATType, "FAT32", 5))
     g_FATParms.FATType              = PXA_FAT_TYPE_32;
else
{
     PXA_ShowDebugMessage("ERROR: Unknown file system (%s).\r\n", FATType);
     g_FATParms.FATType              = PXA_FAT_TYPE_NULL;
     return(FALSE);
}
     // Some BIOS/floppy combinations don't support multi-sector reads.得到一
次可以读取多少数据
     ReadBufferLen = ((pBPB->FatEx.Fat16Ex.DriveId < 0x80) ?
pBPB->BytesPerSect : READ_BUFFER_LENGTH_MAX);
//因为SDMMC卡类型=0 当作软盘使用,因此在此需要不管什么全部取最大值
     ReadBufferLen = READ_BUFFER_LENGTH_MAX;

PXA_ShowDebugMessage("Info: DriveId=0x%x FATType=%s\r\n",
pBPB->FatEx.Fat16Ex.DriveId ,FATType);

//
// Initialize the read buffer.
//g_FileReadBuffer
if (!InitReadBuffer((ULONG)g_FileReadBuffer, ReadBufferLen))
{
     PXA_ShowDebugMessage("ERROR: Failed to initialize read buffer.\r\n");
     return(FALSE);
}


//FAT所在的扇区序号
g_FATParms.FATSector = pBPB->RsvdSects + nReSectors;

//FAT16文件系统根目录所在的扇区位置 ,根目录占用32扇区
PXA_ShowDebugMessage("INFO: RSVD=0x%x FATS=0x%x SPF=0x%x  RS=0x%x\r\n",
     pBPB->RsvdSects, pBPB->NumFATs,pBPB->SectsPerFAT,nReSectors);

g_FATParms.RootDirSector = pBPB->RsvdSects + pBPB->NumFATs *
```

```
pBPB->SectsPerFAT + nReSectors;

  RootDirSectors =
( (pBPB->NumRootEntries*32)+(pBPB->BytesPerSect-1) )/pBPB->BytesPerSect;
  //数据开始扇区
  g_FATParms.DataStartSector = pBPB->RsvdSects + pBPB->NumFATs *
pBPB->SectsPerFAT + RootDirSectors + nReSectors;

  PXA_ShowDebugMessage("INFO: RootDirSector=0x%x DataStartSector=0x%x\r\n",
      g_FATParms.RootDirSector,g_FATParms.DataStartSector);

  return TRUE;
  }

  ///----------------------------------------
  ///@brief 2. 打开文件
  /// Open filename specified and return the size.
  ///@param pFileName = 文件名
  ///@return 返回实际的文件长度
  ///----------------------------------------
  ULONG PXA_OpenFile(PCHAR pFileName)
  {
      USHORT i = 0;
      USHORT j = 0;
      BOOL bFound = FALSE;
  ULONG DirSector = 0;//根目录扇区序号
      CHAR Sector[SECTOR_SIZE];
      PPXA_DIRENTRY  pDirEntry = NULL;//目录结构
      CHAR FileName[11] = {' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' '};

  //
  // Check caller's parameters.
  //
      if (pFileName == NULL)
          return(0);
  //
  // Convert the filename to the form stored on disk.
  //
      for (i = 0 ; i < 8 && *(pFileName + i) != '\0' && *(pFileName + i) != '.' ;
i++)
          FileName[i] = TO_UPPER(*(pFileName + i));

      if (*(pFileName + i) == '.')
      {
          i++;
          for (j=0 ; i < 12 && *(pFileName + i) != '\0' ; i++, j++)
          FileName[8 + j] = TO_UPPER(*(pFileName + i));
      }

  //
```

```
    // Look for the filename in directory list.
    //
    for (bFound = FALSE, i = 0, DirSector = g_FATParms.RootDirSector ; !bFound
&& i < pBPB->NumRootEntries ; DirSector++)
    {
        memset(&Sector[0], 0, SECTOR_SIZE);
        PXA_ShowDebugMessage("INFO: Reading root directory sector =0x%x ,
RDS=0x%x...\r\n", DirSector,g_FATParms.RootDirSector);
    //
    // Read a sector from the root directory.
    //
        if (!PXA_ReadSectors(DirSector, 1, Sector))//读1个扇区的数据
        {
            PXA_ShowDebugMessage("ERROR: Couldn't read root directory
sector=0x%x. \r\n", DirSector);
            return(0);
        }
    ShowBuffer(Sector);

    PXA_WaitPushKey(0x20);

    //
    // Try to find the specified file in the root directory...
    //
        for (pDirEntry = (PPXA_DIRENTRY)Sector, j = 0 ; j < (pBPB->BytesPerSect
/ sizeof(PXA_DIRENTRY)) && i < pBPB->NumRootEntries ; j++, i++, pDirEntry++)
        {
        if( pDirEntry->FileAttr==0x0F ) continue;//长文件名，不支持
        if( pDirEntry->FileAttr==0x10 ) continue;//目录，不支持
        if(pDirEntry->FileAttr==0x08 ) continue;//卷标，不支持

        PXA_ShowDebugMessage("INFO: %d: File name '%s' File size = 0x%x\r\n",
j, pDirEntry->FileName, pDirEntry->FileSize);
        PXA_WaitPushKey(0x20);
            if (!memcmp(FileName, pDirEntry->FileName, 11))
            {
            //
            // Found it.
            //
                bFound = TRUE;
                break;
            }
        }
    }

    if (!bFound)
    {
        PXA_ShowDebugMessage("ERROR: Couldn't find file '%s'.\r\n",
pFileName);
        return(0);
    }
```

```
    else
    {
        PXA_ShowDebugMessage("INFO: Found file '%s' (start cluster = 0x%x size
= 0x%x)\r\n", pFileName, pDirEntry->FirstClust, pDirEntry->FileSize);
    }

  //
    // Save file parameters
  //
    g_FileInfo.FileSize            = pDirEntry->FileSize;
  g_FileInfo.FirstCluster          = pDirEntry->FirstClust;
  g_FileInfo.CurCluster            = pDirEntry->FirstClust;
  g_FileInfo.NumContigClusters     = 0;//相邻文件簇个数
    g_FileInfo.pCurReadBuffAddr      = g_pReadBuffStart;//当前读指针地址
  g_FileInfo.NumReadBuffBytes      = 0;//缓冲区还有的数据字节个数

    return(g_FileInfo.FileSize);
}

///---------------------------------------
///@brief 3. 关闭文件
///---------------------------------------
void PXA_CloseFile(void)
{
    memset(&g_FileInfo, 0, sizeof(g_FileInfo));
}


//-------------------------------
///@brief 是否为数据簇
//-------------------------------
BOOL IsDataCluster(ULONG Cluster)
{
 switch(g_FATParms.FATType)
 {
 case PXA_FAT_TYPE_12:
     if (Cluster >= 0x002 && Cluster <= 0xfef)
         return(TRUE);
     break;
 case PXA_FAT_TYPE_32:
     Cluster &= 0x0fffffff;
     if (Cluster >= 0x00000002 && Cluster <= 0x0ffffef)
         return(TRUE);
     break;
 case PXA_FAT_TYPE_16:
 default:
     if (Cluster >= 0x0002 && Cluster <= 0xffef)
         return(TRUE);
 }

 return(FALSE);
}
```

```
//-------------------------------
///@brief 是否为保留文件簇
//-------------------------------
BOOL IsRsvdCluster(ULONG Cluster)
{
 switch(g_FATParms.FATType)
 {
 case PXA_FAT_TYPE_12:
     if (Cluster >= 0xff0 && Cluster <= 0xff6)
         return(TRUE);
     break;
 case PXA_FAT_TYPE_32:
     Cluster &= 0x0fffffff;
     if (Cluster >= 0x0ffffff0 && Cluster <= 0x0ffffff6)
         return(TRUE);
     break;
 case PXA_FAT_TYPE_16:
 default:
     if (Cluster >= 0xfff0 && Cluster <= 0xfff6)
         return(TRUE);
 }

 return(FALSE);
}

//-------------------------------
///@brief 是否为结束文件簇
//-------------------------------
BOOL IsEOFCluster(ULONG Cluster)
{
 switch(g_FATParms.FATType)
 {
 case PXA_FAT_TYPE_12:
     if (Cluster >= 0xff8 && Cluster <= 0xfff)
         return(TRUE);
     break;
 case PXA_FAT_TYPE_32:
     Cluster &= 0x0fffffff;
     if (Cluster >= 0x0ffffff8 && Cluster <= 0x0fffffff)
         return(TRUE);
     break;
 case PXA_FAT_TYPE_16:
 default:
     if (Cluster >= 0xfff8 && Cluster <= 0xffff)
         return(TRUE);
 }

 return(FALSE);
}
```

```
//-------------------------------
///@brief 是否为坏文件簇
//-------------------------------
BOOL IsBadCluster(ULONG Cluster)
{
 switch(g_FATParms.FATType)
 {
 case PXA_FAT_TYPE_12:
     if (Cluster == 0xff7)
         return(TRUE);
     break;
 case PXA_FAT_TYPE_32:
     Cluster &= 0x0fffffff;
     if (Cluster == 0x0ffffff7)
         return(TRUE);
     break;
 case PXA_FAT_TYPE_16:
 default:
     if (Cluster == 0xfff7)
         return(TRUE);
 }

 return(FALSE);
}

//-------------------------------
///@brief 得到下一个文件簇
//-------------------------------
ULONG GetNextCluster(ULONG Cluster)
{
 ULONG Sector = 0;
 ULONG ByteOffset = 0;
 PUCHAR pSectorCache = (PUCHAR)g_FATCACHE;//Sector cache is where the sector
used to read the FAT cluster chains lives.
 static ULONG CurrentSector = 0;
 ULONG NextCluster = 0;
 //PXA_ShowDebugMessage("..........Cluster=0x%x \n",Cluster);
 // If we're passed an EOF cluster, return it.
 //
 if (IsEOFCluster(Cluster))//是最后一个文件簇则直接返回
        return(Cluster);

 // Is caller giving us a valid cluster?
 //PXA_ShowDebugMessage("..........Cluster=0x%x \n",Cluster);

 //
    if (!IsDataCluster(Cluster))//判断是否为数据文件簇
 {
        PXA_ShowDebugMessage("ERROR: GetNextCluster - bad cluster
number.\r\n");
     return(0);  // 0 isn't a valid cluster number (at least for our purposes).
```

```
}

// Compute sector where our FAT entry lives.
//
switch(g_FATParms.FATType)
{
case PXA_FAT_TYPE_12:
    Sector = (Cluster * 3) / 2;     // Every FAT12 cluster is 1.5 bytes.
    ByteOffset = Sector % pBPB->BytesPerSect;
    Sector /= pBPB->BytesPerSect;
    Sector += g_FATParms.FATSector;
    break;
case PXA_FAT_TYPE_32:
    //ByteOffset = 4;
    PXA_ShowDebugMessage("ERROR: FAT 32 isn't currently supported.\r\n");
    return(0);
    break;
case PXA_FAT_TYPE_16:
default:
    Sector = Cluster * sizeof(USHORT);
    ByteOffset = Sector % pBPB->BytesPerSect;
    Sector /= pBPB->BytesPerSect;
    Sector += g_FATParms.FATSector;
}

// If the sector we're interested in isn't in our cache, get it.
//判断是否应该读FAT分配表数据
if (CurrentSector != Sector)
{
    //读FAT分配表到 pSectorCache
    if (!PXA_ReadSectors(Sector, 1, pSectorCache))
    {
        PXA_ShowDebugMessage("ERROR: GetNextCluster - unable to read
sector.\r\n");
    }
//  PXA_ShowDebugMessage("Info: Read FATSector to pSectorCache.\r\n");
//  ShowBuffer(pSectorCache);

//  PXA_WaitPushKey(0x0D);
//  PXA_ShowBuffer
    CurrentSector = Sector;
}

// Locate next cluster number...
//
switch(g_FATParms.FATType)
{
case PXA_FAT_TYPE_12:
    // Need to worry about cluster number crossing a sector boundary.
    if (ByteOffset == ((ULONG)pBPB->BytesPerSect - 1))
    {
        NextCluster = (ULONG)(*(PUCHAR)(pSectorCache + ByteOffset));
```

```
            // Now we need to read the next sector.
            //
            ++Sector;
            if (!PXA_ReadSectors(Sector, 1, pSectorCache))
            {
                PXA_ShowDebugMessage("ERROR: GetNextCluster - unable to read
sector.\r\n");
            }
            CurrentSector = Sector;

            NextCluster |= (ULONG)((*(PUCHAR)pSectorCache) << 8);
        }
        else
            NextCluster = (ULONG)(*(PUSHORT)(pSectorCache + ByteOffset));
        // Since every FAT12 entry is 1.5 bytes, we either need to shift or mask
based
        // on whether the previous cluster number was odd or even.
        //
        if (Cluster & 0x1)
            NextCluster = NextCluster >> 4;
        else
            NextCluster = NextCluster & 0xfff;
        break;
    case PXA_FAT_TYPE_32:
        // FAT32 is easy - no worries about sector boundaries...
        NextCluster = (ULONG)(*(PULONG)(pSectorCache + ByteOffset));
        break;
    case PXA_FAT_TYPE_16:
    default:
        // FAT16 is easy - no worries about sector boundaries...
        {
            NextCluster = (ULONG)(*(PUSHORT)(pSectorCache + ByteOffset));
        //  PXA_ShowDebugMessage("INFO: pSectorCache=0x%x  ByteOffset=0x%x
NextCluster=0x%x \r\n", pSectorCache[ByteOffset], pSectorCache[ByteOffset+1],
NextCluster);
        }
    }

    // PXA_ShowDebugMessage("INFO: GetNextCluster - cluster=0x%x  next
cluster=0x%x.\r\n", Cluster, NextCluster);

    // Return the next cluster value.
    //
    return(NextCluster);
}


//--------------------------------
///@brief 得到下一组连续的文件簇
//--------------------------------
```

```
CLUSTTYPE GetNextClusterGroup(ULONG StartCluster, PULONG pNumberClusters)
{
 ULONG CurrentCluster = 0;
 ULONG NextCluster = 0;

     *pNumberClusters = 0;

 // TODO - could be a bad cluster - check for it here.

 // Determine number of contiguous clusters.
 //
 NextCluster = StartCluster;
 do
 {
     //PXA_ShowDebugMessage("GetNextClusterGroup StartCluster=0x%x
NumClusters=0x%x \n",StartCluster,*pNumberClusters);
     CurrentCluster = NextCluster;
     *pNumberClusters = *pNumberClusters + 1;

         NextCluster = GetNextCluster(CurrentCluster);
     if (NextCluster != (CurrentCluster + 1))
         break;
 } while(IsDataCluster(NextCluster));

 // Why did we break out of the above loop?
 //
 if (IsEOFCluster(NextCluster))
     return(EOF_CLUSTER);
 if (IsBadCluster(NextCluster))
     return(BAD_CLUSTER);
 if (IsRsvdCluster(NextCluster))
     return(RSVD_CLUSTER);

 return(DATA_CLUSTER);
}


///-------------------------------------------
///@brief 4. 读文件
///@param pAddress = 缓冲区地址
///@param Length = 长度
///-------------------------------------------
BOOL PXA_ReadFile(PUCHAR pAddress, ULONG Length)
{
 ULONG  ClustersToRead = 0;

 // Validate callers parameters.
 //
 if (!pAddress || Length == 0)
     return(FALSE);
```

```
    // Make sure callers called InitReadBuffer before calling us.
    //
        if (!g_pReadBuffStart || g_ReadBuffLenInClusters == 0)
    {
        PXA_ShowDebugMessage("ERROR: InitReadBuffer must be called before reading
a file.\r\n");
        return(FALSE);
    }

    do
    {
        // Is there data in the read buffer?
        //
          if (g_FileInfo.NumReadBuffBytes)//判断是否有数据在缓冲区
        {
          if (Length <= g_FileInfo.NumReadBuffBytes)
          {
              // We can satisfy the read request with what's in the read buffer
already.
              //
              memcpy(pAddress, g_FileInfo.pCurReadBuffAddr, Length);
              g_FileInfo.NumReadBuffBytes -= Length;
              g_FileInfo.pCurReadBuffAddr  += Length;
              pAddress                     += Length;
              return(TRUE);
          }
          else
          {
              // Empty the read buffer before loading more.
              //
              memcpy(pAddress, g_FileInfo.pCurReadBuffAddr,
g_FileInfo.NumReadBuffBytes);
              Length   -= g_FileInfo.NumReadBuffBytes;
              pAddress += g_FileInfo.NumReadBuffBytes;
          }
        }

        //
        // Load the entire read buffer with more data.
        //

        // Determine next grouping of clusters that are contiguous (on the storage
device).
        if (g_FileInfo.NumContigClusters == 0)//相邻的文件簇
        {
          // If the current cluster isn't the first cluster in the file, then
the value is +1 from the last
          // cluster in the contiguous grouping.  Subtract one, follow the link
and seek a new grouping.
          //
          if (g_FileInfo.CurCluster != g_FileInfo.FirstCluster)//当前文件簇是否
等与开始文件簇
```

```
    {
        --g_FileInfo.CurCluster;
        if (IsEOFCluster(g_FileInfo.CurCluster))
        {
            //达到了最后文件簇
        //  PXA_ShowDebugMessage("INFO: Reached EOF cluster in FAT file
chain.\r\n");
            return(FALSE);
        }

    //      PXA_ShowDebugMessage("INFO: 相邻的文件簇
CurCluster=0x%x\r\n",g_FileInfo.CurCluster);
        g_FileInfo.CurCluster = GetNextCluster(g_FileInfo.CurCluster);
    //      PXA_ShowDebugMessage("INFO: 下一个文件簇
CurCluster=0x%x\r\n",g_FileInfo.CurCluster);

    }

    // Get the next contiguous grouping of cluster and return the type of
the last cluster in the list.
    //确定有多少可以连续的文件簇
    switch(GetNextClusterGroup(g_FileInfo.CurCluster,
&g_FileInfo.NumContigClusters))
    {
    case DATA_CLUSTER:
        PXA_ShowDebugMessage("INFO: 0x%x contiguous clusters starting
with 0x%x.\r\n", g_FileInfo.NumContigClusters, g_FileInfo.CurCluster);
        break;
    case EOF_CLUSTER:
        PXA_ShowDebugMessage("INFO: 0x%x contiguous clusters starting
with 0x%x (EOF terminator).\r\n", g_FileInfo.NumContigClusters,
g_FileInfo.CurCluster);
        break;
    case RSVD_CLUSTER:
        PXA_ShowDebugMessage("ERROR: Found reserved cluster in FAT file
chain.\r\n");
        return(FALSE);
        break;
    case BAD_CLUSTER:
        PXA_ShowDebugMessage("ERROR: Found bad cluster in FAT file
chain.\r\n");
        return(FALSE);
        break;
    default:
        PXA_ShowDebugMessage("ERROR: Unable to determine contiguous
cluster grouping (start cluster = 0x%x).\r\n", g_FileInfo.CurCluster);
        return(FALSE);
    }
  }

  // Determine the number of clusters we can read at this time.
  //
```

```
    ClustersToRead = MIN(g_ReadBuffLenInClusters,
g_FileInfo.NumContigClusters);//决定这次读多少连续的文件簇

    ///----------------------------------------------------------------
    ///@brief 3. 读指定扇区的数据
    ///@param dwSectorIndex = 扇区开始序号
    ///@param nSecCount = 扇区个数
    ///@param pBlockBuffer = 块缓冲区地址
    ///----------------------------------------------------------------
    //BOOL  PXA_ReadSectors(DWORD dwSectorIndex,WORD nSecCount ,BYTE*
pBlockBuffer)

    // Read data from storage.
    //从设备读具体数据
  //  PXA_ShowDebugMessage("INFO :当前文件簇=0x%x
\r\n",g_FileInfo.CurCluster);
  //  PXA_WaitPushKey(0x20);
    {
    //在此需要将文件簇号转换成对应的扇区序号
        DWORD dwSecIndex = 0x00;
        dwSecIndex = g_FATParms.DataStartSector + (g_FileInfo.CurCluster -
2)* pBPB->SectsPerClust ;

        if (!PXA_ReadSectors(dwSecIndex , (USHORT)(ClustersToRead *
pBPB->SectsPerClust), g_pReadBuffStart))
        {
            PXA_ShowDebugMessage("ERROR: Failed to read data from storage
(starting cluster = 0x%x  number of clusters = 0x%x).\r\n", g_FileInfo.CurCluster,
ClustersToRead);
            return(FALSE);
        }
    }
    // Update read management data.
    //
        g_FileInfo.NumContigClusters -= ClustersToRead;//改变现有的相邻文件簇个
数
    g_FileInfo.CurCluster           += ClustersToRead;//当前文件簇序号
    g_FileInfo.pCurReadBuffAddr      = g_pReadBuffStart;//当前地址
    g_FileInfo.NumReadBuffBytes      = (ClustersToRead * pBPB->SectsPerClust *
pBPB->BytesPerSect);//缓冲区有的字节数据
    ////////////////////////
    //在此更新系统显示信息。
    {
        static  ULONG   st_CurrentFileSize = 0x00;
        DWORD   dwOffset = 0x00;
        st_CurrentFileSize += g_FileInfo.NumReadBuffBytes;

        dwOffset = (st_CurrentFileSize*100) / g_FileInfo.FileSize;
        dwOffset = MIN(dwOffset,100);
        PXA_ShowGuage(dwOffset);
```

```
        }

        ////////////////////////
    //  PXA_ShowDebugMessage("g_FileInfo.CurCluster=0x%x\r\n",
g_FileInfo.CurCluster);

        }
    while (Length);

    return(TRUE);
    }

    static BOOL InitReadBuffer(ULONG StartAddress, ULONG Length)
    {
    // Check arguments.
    //
    if (StartAddress == 0)
        return(FALSE);

    // Determine how many clusters are in the read buffer.
    //
    Length /= (pBPB->BytesPerSect * pBPB->SectsPerClust);
    if (!Length)
    {
        PXA_ShowDebugMessage("ERROR: Read buffer should be at least a cluster size
in length.\r\n");
        return(FALSE);
    }

    // Store read buffer start and the adjusted length.
    //
    g_pReadBuffStart = (PUCHAR)StartAddress;//缓冲区地址
    g_ReadBuffLenInClusters = Length;//缓冲区包含的文件簇为单位的数据长度

    return(TRUE);
    }

    ///-----------------------------------------------------------
    /// Load a BIN file into memory and jump to the start address.
    ///@brief 5. 装入BIN文件
    ///@param pFileName = BIN文件名称
    ///@param pImageLoc = 返回的BIN文件入口地址
    ///-----------------------------------------------------------
```

```
BOOL PXA_LoadBINFile(PCHAR pFileName, PULONG pImageLoc)
{
    IMAGEHDR ImageHdr;
    RECORDHDR RecordHdr;

    //
    // Check callers arguments
    //
    if (!pFileName || !pImageLoc)
        return(FALSE);

  *pImageLoc = 0;

    //
    // Open the BIN file for reading
    //
    if (!PXA_OpenFile(pFileName))
    {
        PXA_ShowDebugMessage("ERROR: File open failed (%s).\r\n", pFileName);
        return(FALSE);
    }

  //
    // Read the BIN file header
    //
    if (!PXA_ReadFile( (PCHAR)&ImageHdr, sizeof(IMAGEHDR)) )
    {
        PXA_ShowDebugMessage("ERROR: Couldn't read image header.\r\n");
        return(FALSE);
    }

  //
    // Display image header
  //
    PXA_ShowDebugMessage("\r\nImage Header:\r\n");
  PXA_ShowDebugMessage("------------------------------------------------\r\n
");
    PXA_ShowDebugMessage("Sync Bytes ................. '%c%c%c%c%c%c'\r\n",
ImageHdr.SyncBytes[0], ImageHdr.SyncBytes[1], ImageHdr.SyncBytes[2],
ImageHdr.SyncBytes[3], ImageHdr.SyncBytes[4], ImageHdr.SyncBytes[5]);
    PXA_ShowDebugMessage("Image Address ............. 0x%x\r\n",
ImageHdr.ImageAddr);
    PXA_ShowDebugMessage("Image Length .............. 0x%x\r\n",
ImageHdr.ImageLen);

    do
    {
        //
        // Read the next BIN file record header
        //
```

```
        if (!PXA_ReadFile((PCHAR)&RecordHdr, sizeof(RECORDHDR)))
        {
            PXA_ShowDebugMessage("ERROR: Couldn't read record header.\r\n");
            return(FALSE);
        }

        //
            // Display Record Header
        //
            PXA_ShowDebugMessage("Record Header:\r\n");

PXA_ShowDebugMessage("------------------------------------------------\r\n");
            PXA_ShowDebugMessage("Record Address ......................
0x%x\r\n", RecordHdr.RecordAddr);
            PXA_ShowDebugMessage("Record Length .......................
0x%x\r\n", RecordHdr.RecordLen);
            PXA_ShowDebugMessage("Record CheckSum .....................
0x%x\r\n", RecordHdr.RecordChksum);

        //
            // Last Header?
        //
            if (RecordHdr.RecordAddr == 0 && RecordHdr.RecordChksum == 0)
        {
                // The last record header contains the image start address (in the
length field).
             //
                *pImageLoc = RecordHdr.RecordLen;
                break;
        }

            //
            // Read the entire record into RAM.
            //
        if (!PXA_ReadFile((PUCHAR)RecordHdr.RecordAddr, RecordHdr.RecordLen))
        {
            PXA_ShowDebugMessage("ERROR: Couldn't read record information
(address=0x%x).\r\n", RecordHdr.RecordAddr);
            return(FALSE);
        }

       }
       while(1);

   //
       // Close file
   //
       PXA_CloseFile();

       return(TRUE);
   }
```

定义 SD 卡读写驱动的公共结构如下所示，对应的文件名为 PXA_SDMMC.H。

```
//-------------------------------------------------------
//描述：PXA270 访问SD/MMC卡程序
//日期：2007-02-09
//修改记录：
//-------------------------------------------------------
#include <windows.h>
#include <nkintr.h>
#include <bulverde.h>
#include <mainstoneii.h>
#include <oal_memory.h>
#include <pcireg.h>
#include <fmd.h>
#include <xllp_pccardsocket.h>
#include <bsp.h>
#include "loader.h"
#include <xllp_mmc.h>
#include "PXA_SDMMC.H"

///////////////////////////////////////////
///@brief 外部引入函数定义
//extern msWait(unsigned msVal);
//extern int OEMReadDebugByte();
//extern void EdbgOutputDebugString( const unsigned char *sz, ...);
extern void ShowBuffer(BYTE* pBlock);
extern void  Launch(unsigned int uAddr);



///@brief 内部全局数据指针定义
static   P_XLLP_GPIO_T              g_pPXA_GpioReg = NULL;
static   P_XLLP_CLKMGR_T            g_pPXA_ClckReg = NULL;
static   P_XLLP_MMC_T               g_pPXA_MmcReg = NULL;
static   UINT32                     g_PXA_SDMMCOCR = 0x00;//OCR寄存器
static   WORD                       g_PXA_ISSDCARD = 0x00;//是否为SD卡 MMC= 1
SD = 2;
static   DWORD                      g_PXA_SD_ADDRESS = 0x00;//SD卡地址，因为SD
卡地址是通过CMD3返回的，这和MMC不同

///@brief 系统定义的SDMMC卡地址，而且系统只支持一个卡
//#define PXA_SDMMC_ADDR          0x200000
#define PXA_SDMMC_ADDR        0x420000
#define PXA_BLOCK_SIZE        0x000200     //扇区的大小=512字节


//----------------------------------------------------------------------
//FILEDIRECTORY;//文件目录结构
static   UINT32          g_NKFindFlags = 0x00;//是否找到对应的文件标志
static   FILEDIRECTORY   g_NK_INI_DIR;//NK   配置文件 NK.INI 目录信息
static   FILEDIRECTORY   g_NK_BIN_DIR;//NK   镜像文件 NK.BIN 目录信息
```

```
static DWORD         g_BiosReSectors = 0x00;//相对扇区数(Relative Sectors)
从该磁盘的开始到该分区的开始的位移量，以扇区来计算
static  DWORD        g_DataSectorStart = 0x00;//文件数据扇区开始位置。由得到
FAT分区类型时得到
static  WORD         g_SectorsPerClust = 0x00;//文件簇包含的扇区个数
static  BYTE         g_PXA_FAT[512] = { 0x00 };//文件分配表的第一个扇区内容
static  PXA_FAT_TYPE g_PXA_FAT_TYPE;//  FAT类型

#define FIND_NULL_FLAGS          0x00000000  //没有
#define FIND_NKBIN_FLAGS         0x00000001  //找到NK.BIN标志
#define FIND_NKINI_FLAGS         0x00000002  //找到NK.INI标志

//------------------------------------------------------------------

typedef struct {
    UINT SCR_STRUCTURE : 4;
    UINT SD_SPEC : 4;
    UINT DATA_STAT_AFTER_ERASE : 1;
    UINT SD_SECURITY : 3;
    UINT SD_BUS_WIDTH : 4;
    UINT RES2 : 16;
    UINT RES1 : 32; // RESERVED FOR MANUFACTURER
}SCR;//SCR scr;

typedef struct
{
UINT CMD;
UINT ADDR;
UINT FSIZE;
UINT BSIZE;
UINT RATE;
UINT NOB;
UINT DATATRANS;
BOOL BUSY;
BOOL READSCR;
}mmcStack;

typedef enum
{
  DNONE=0,
  WRITE,
  READ
}dataDirection;

typedef struct{
UINT32 lresponse[4];
UINT32 sresponse;
UINT32 status;
}STATRES;
typedef struct {
    UINT SZ_PROC : 32;
```

```
    UINT SD_CARD_TYPE : 16;
    UINT RES2 : 13;
    UINT SECD_MODE : 1;
    UINT DAT_BUS_WIDTH : 2;
} STATUS;
typedef struct {
        UINT ECC : 2;                    // Bits 0-31
        UINT RES1 : 2;
        UINT TMP_WRITE_PROTECT : 1;
        UINT PERM_WRITE_PROTECT : 1;
        UINT COPY : 1;
        UINT RES2 : 1;
        UINT RES3 : 5;
        UINT WRITE_BL_PARTIAL : 1;
        UINT WRITE_BL_LEN : 4;
        UINT R2W_FACTOR : 3;
        UINT DEFAULT_ECC : 2;
        UINT WP_GRP_ENABLE : 1;
        UINT WP_GRP_SIZE : 5;
        UINT ERASE_GRP_MULT : 3;

        UINT ERASE_GRP_MULTx : 2;    // Bits 32-63
        UINT ERASE_GRP_SIZE : 5;
        UINT C_SIZE_MULT : 3;
        UINT VDD_W_CURR_MAX : 3;
        UINT VDD_W_CURR_MIN : 3;
        UINT VDD_R_CURR_MAX : 3;
        UINT VDD_R_CURR_MIN : 3;
        UINT C_SIZE : 10;

        UINT C_SIZEx : 2;            // Bits 64-95
        UINT RES4 : 2;
        UINT DSR_IMP : 1;
        UINT READ_BLK_MISALIGN : 1;
        UINT WRITE_BLK_MISALIGN : 1;
        UINT READ_BL_PARTIAL : 1;
        UINT READ_BL_LEN : 4;
        UINT CCC : 12;
        UINT TRAN_SPEED : 8;

        UINT NSAC : 8;               // Bits 96-127
        UINT TAAC : 8;
        UINT RES5 : 2;
        UINT SPEC_VERS : 4;
        UINT CSD_STRUCTURE : 2;
        UINT RES6 : 8;
}MMCCSD;
typedef struct {
    UINT NU : 1;     // 0- 31
    UINT CRC : 7;
    UINT MDT : 8;
    UINT PSN : 16;
```

```
    UINT PSNx : 16;  // 32- 63
    UINT PRV : 8;
    UINT PNM : 8;

    UINT PNMx :32;    // 64- 96

    UINT PNMxx : 7;   //96 - 127
    UINT OID : 16;
    UINT MID : 8;
}MMCCID;

typedef struct{
 UINT32 block;
 UINT32 *buffer;
 UINT size;
 BOOL mmc;
 UINT32 OCR;
 UINT32 rca;
 MMCCSD *mmc_csd;
 MMCCID *mmc_cid;
 SCR *sd_scr;
 STATUS sd_status;
}MMCSDAttribT;

#define SO_SCR          0x8;  // size of sd scr register
#define SO_BLOCK        0x200;
#define RCA_MMC_CARD    0x420000;
#define CMD1_ADDR       0x200000;

#define MMCSD_CARD_STATUS_OOR      (1u<<31)     // out of range
#define MMCSD_CARD_STATUS_AE       (1u<<30)    // address error
#define MMCSD_CARD_STATUS_EP       (1u<<27)    // erase param
#define MMCSD_CARD_STATUS_WPV      (1u<<26)    // write protected block
#define MMCSD_CARD_STATUS_CL       (1u<<25)    // card locked
#define MMCSD_CARD_STATUS_ERR      (1u<<19)    // general error
#define MMCSD_CARD_STATUS_CS       (0xfu<<9)   // current state
#define MMCSD_CARD_STATUS_RFD      (1u<<8)     // ready for data
#define MMCSD_CARD_STATUS_AC       (1u<<5)     // app command

#define MMCSD_CARD_OCR_nBUSY       (1u<<31)      // app command

// this is the structure of the status response defined in the LLD

#define  MMCSD_STAT_TOR                  XLLP_STAT_TORD
#define  MMCSD_STAT_TORSPN               XLLP_STAT_TORSPNS
#define  MMCSD_STAT_CRCWRER              XLLP_STAT_CRCWRERR
#define  MMCSD_STAT_CRCRDERR             XLLP_STAT_CRCRDERR

#define  MMCSD_STAT_DATAERRTKN           XLLP_STAT_DATAERRTKN
#define  MMCSD_STAT_RESCRCERR            XLLP_STAT_RESCRCERR
#define  MMCSD_STAT_FLASHERR             (1u<<6)
```

```
#define  MMCSD_STAT_CARD_ERR              (1u<<8)
#define  MMCSD_STAT_CARD_LOCKED           (1u<<9)
#define  MMCSD_STAT_WP_VIOLATION          (1u<<10)
#define  MMCSD_STAT_ERASE_PARAM           (1u<<11)
#define  MMCSD_STAT_ADDRESS_ERR           (1u<<12)
#define  MMCSD_STAT_OUT_OF_RANGE          (1u<<13)
#define  MMCSD_STAT_CARD_BUSY             (1u<<14)
#define  MMCSD_STAT_CURRENT_CARD_STATE   (0xfu<<28)


//-------------------------------------------------------------------------------------------
/*
static void PXA_ShowDebugMessage(const unsigned char *sz, ...)
{
}

*/
///@brief 4.显示调试信息
#define PXA_ShowDebugMessage KITLOutputDebugString

void WaitSpace()
{
//   while(0x20!=OEMReadDebugByte());
}

///@brief 测试SDMMC卡
void TestSDMMC()
{
//   int index;
 int size=0;
 BOOL    bSD = FALSE;

 mmcStack         MMCSTK;
 STATRES          mmcsdStatus;
 MMCSDAttribT     mmcsdAttribT;
//   MMCCID           mmccid;
//   MMCCSD           mcccsd;
 XLLP_CLKMGR_T    *clkMgr = NULL;//(XLLP_CLKMGR_T*) CLK_BASE_U_VIRTUAL;
 XLLP_GPIO_T      *pGPIO = NULL;
 XLLP_MMC_T       *pMMC = NULL;//(XLLP_MMC_T* ) MMC_BASE_U_VIRTUAL;
    PXA_ShowDebugMessage ("\r\n 测试SDMMC卡 ...\r\n");

 pGPIO   = (XLLP_GPIO_T *) OALPAtoVA(BULVERDE_BASE_REG_PA_GPIO,
FALSE);//FALSE=U TUE=V
 pMMC    = (XLLP_MMC_T *) OALPAtoVA(BULVERDE_BASE_REG_PA_MMC, FALSE);//FALSE=U
TUE=V
 clkMgr = (XLLP_CLKMGR_T *) OALPAtoVA(BULVERDE_BASE_REG_PA_CLKMGR,
FALSE);//FALSE=U TUE=V
```

```
    PXA_InitSDMMC(pGPIO,pMMC,clkMgr);

    PXA_CMD0();

    if(PXA_IsSDCard())
    {
            PXA_ShowDebugMessage ( "\r\n 是SD卡...\r\n");
        bSD = TRUE;
        PXA_CMD55();
        {
            PXA_STATRES temp =  PXA_ACMD41();
            mmcsdStatus = *((STATRES*)&temp);
        }
        mmcsdAttribT.OCR  = mmcsdStatus.sresponse;
    }
    else
    {
        PXA_STATRES temp = PXA_CMD1();
        mmcsdStatus = *((STATRES*)&temp);
        if( mmcsdStatus.status == XLLP_STAT_TORSPNS)//超时
        {
            PXA_ShowDebugMessage ( "\r\n 没有卡 ...\r\n");
            WaitSpace();
            return ;
        }
            PXA_ShowDebugMessage ( "\r\n 发现了MMC卡...\r\n");
        mmcsdAttribT.OCR  = mmcsdStatus.sresponse;
    }


    goto SendCMD2_1;
    ///////////////////////////////////


   SendAppCMD:
    {
        DWORD   MBR_OFFSET = 0x00;//主引导扇区保留
        DWORD   dwFRSI = 0x00;
        MBR_DPT*    pDPT = NULL;
        BYTE buffer[4096];
        pDPT = (MBR_DPT*)&buffer[0x01BE];
        PXA_CMD17(0,buffer);//读MBR
        ShowBuffer(buffer);
        PXA_ShowDebugMessage("BootID=0x%x SID=0x%x RS=0x%x TS=0x%x\r\n",
                pDPT->BootID,pDPT->SystemID,pDPT->ReSectors,pDPT->TotalSec);

        //////////////////////////////////////////
        PXA_ShowDebugMessage("读 BIOSPB 扇区\r\n");
        WaitSpace();
        MBR_OFFSET = pDPT->ReSectors;
        PXA_CMD17(pDPT->ReSectors, buffer);
```

```
        ShowBuffer(buffer);
        //BIOSPB PXA_FAT_TYPE CheckFATType(BIOSPB* pBPB, DWORD
*dwFirstRootDirSecIndex, WORD* nSectsPerClust, WORD* nFATSecIndex)
        {
            PXA_FAT_TYPE   PFT;
            DWORD   dwFirstRootDirSecIndex;
            WORD    nSectsPerClust;
            WORD    nFATSecIndex;
            PFT =
CheckFATType((BIOSPB*)buffer,&dwFirstRootDirSecIndex,&nSectsPerClust,&nFATSecI
ndex);
            PXA_ShowDebugMessage("PFT= %d FRSI=0x%x   SPC=0x%x
FSI=0x%x\r\n",
                PFT,dwFirstRootDirSecIndex,nSectsPerClust,nFATSecIndex);
            ///////////////////////////////////////////////////////////////
            dwFRSI = dwFirstRootDirSecIndex;
        }
        WaitSpace();

        //初始化文件系统

        PXA_InitFAT(buffer,(WORD)MBR_OFFSET);
        {
            UCHAR* strFileName = "NK.BIN";
            DWORD   dwFileSize = 0x00;
    //          BYTE    MYBuf[0x200];
            WORD    readSize = 0x200;
            DWORD   ReadCount = 0x00;
            DWORD i = 0;

            //
            // Open the BIN file for reading
            //
            {
                ULONG ImageLoc = 0x00;
                ULONG ImagePhLoc = 0x00;
                if(!PXA_LoadBINFile(strFileName , &ImageLoc ))
                {
                    PXA_ShowDebugMessage("ERROR: LoadBinFile Failed .\r\n");
                }
                else
                {
                    ImagePhLoc = (UINT32) OALVAtoPA((VOID *)ImageLoc);
                    PXA_ShowDebugMessage("INFO: LoadBinFile
OK .VStartAddress=0x%x \r\n",ImageLoc);
                    PXA_ShowDebugMessage("INFO: LoadBinFile
OK .PStartAddress=0x%x \r\n",ImagePhLoc);

                }
                ///////////////
```

```
                PXA_ExitSDMMC();
                PXA_ShowDebugMessage("INFO: Push K Key to Launch CE...\r\n");

    /*          while(1)
                {
                    int key =OEMReadDebugByte();
                    if( (key == 'K')||( key =='k') )
                    {
                        break;
                    }

                }
    */
                WaitSpace();
                Launch(ImagePhLoc);

            }
            //PXA_CloseFile();

        }

        return;



        {
            WORD DirOffset = (WORD)(MBR_OFFSET + dwFRSI);
            PXA_ShowDebugMessage("读主目录扇区 0x%x \r\n",DirOffset);
            PXA_CMD17( DirOffset, buffer);
            ShowBuffer(buffer);
            //void  GetRootDirFile(BYTE* pDirSecBuffer)//得到文件目录
            GetRootDirFile(buffer);
            {
                //FIND_NKINI_DIR
                FILEDIRECTORY* pDIRINI = GetFileDir(FIND_NKINI_DIR);
                FILEDIRECTORY* pDIRBIN = GetFileDir(FIND_NKBIN_DIR);
                if( pDIRINI != NULL)
                {
                    PXA_ShowDebugMessage("找到了文件 NK.INI\r\n");
                    PXA_ShowDebugMessage("开始簇号=0x%x 文件长度
=%d\r\n",pDIRINI->CulsterLow,pDIRINI->FileLength);
                }
                if( pDIRBIN != NULL)
                {
                    int j = 0;
                    DWORD i;

                    PXA_ShowDebugMessage("找到了文件 NK.BIN\r\n");
```

```
                    PXA_ShowDebugMessage("开始簇号=0x%x 文件长度
=%d\r\n",pDIRBIN->CulsterLow,pDIRBIN->FileLength);
                    PXA_ShowDebugMessage("\n....按空格键开始读取32M数据...\r\n");
                    WaitSpace();

                    for( i = 0 ; i < 2048*32 ; i++)
                    {
                        PXA_CMD17( DirOffset, buffer);
                    }
                    PXA_ShowDebugMessage("read Data %d M\r\n",32);

                //  ShowBuffer(buffer);

                }
            }
        }



    }
    goto end;

   SendCMD1:
    {
        PXA_STATRES temp ;
        if( !bSD)
        {
            temp = PXA_CMD1();
        }
        else
        {//SD卡
            PXA_CMD55();
            temp = PXA_ACMD41();
        }
        mmcsdStatus = *((STATRES*)&temp);
        mmcsdAttribT.OCR = mmcsdStatus.sresponse;
        if (mmcsdAttribT.OCR & MMCSD_CARD_OCR_nBUSY)  // Card power up status
bit (!busy)  {

            mmcsdAttribT.mmc = bSD ? XLLP_FALSE : XLLP_TRUE;
                MMCSTK.BUSY = XLLP_FALSE;
            PXA_ShowDebugMessage ( "\r\n go to SendCMD2 .广播,得所有卡CID..\r\n");
                goto SendCMD2;
            }
            else
            {
```

```
                PXA_ShowDebugMessage ( "\r\n next go to SendCMD1 ...\r\n");
                goto SendCMD1;
          }
     }

   SendCMD2_1:
    if (!(mmcsdStatus.sresponse & MMCSD_CARD_OCR_nBUSY))  // Card power up
status bit (!busy)
        {
        PXA_ShowDebugMessage ( "\r\n goto SendCMD1...\r\n");
        goto SendCMD1;
        }
        else
        {
            MMCSTK.BUSY = XLLP_FALSE;
            PXA_ShowDebugMessage ( "\r\n SendCMD2_1 go to SendCMD2...\r\n");
            goto SendCMD2;
        }

   SendCMD2://发送广播命令，得到所有卡的CID
   //printf("+SendCMD2\r\n");
        WaitSpace();
        PXA_CMD2();

            PXA_ShowDebugMessage ( "\r\n goto SendCMD3 确认卡地址...\r\n");
            goto SendCMD3;

   SendCMD3://确认卡地址
   //printf("+SendCMD3\r\n");
        {
            WaitSpace();
            PXA_CMD3();
            goto EndInitClock;
        }

   EndInitClock:
   //printf("+EndInitClock\r\n");
   //              XllpMmcSdInts(MmcReg,XLLP_MMC_I_MASK_CLKISOFF);
   //          MmcReg->MMC_STRPCL = XLLP_STRPCL_STPCLK;         // Stop the MMC
clock
    //             osStatus = OSAFlagWait(_keypadFlagRef, KEYPAD_FLAG_MASK,
OSA_FLAG_OR_CLEAR, &keypadFlags, OSA_SUSPEND);
   //          ASSERT(osStatus == OS_SUCCESS);
   //              while (MmcReg->MMC_STAT & XLLP_STAT_CLKEN);    // Wait for
the clock to stop
   //          XllpMmcSetUpClock(MmcReg, XLLP_MMC_19_5MHz, 1);
            PXA_ShowDebugMessage ( "\r\n goto SendCMD9 确认卡CSD...\r\n");
            goto SendCMD9;

   SendCMD7://选择确认地址的卡
            WaitSpace();
```

```
        PXA_CMD7();
        ////////////////////
        PXA_ShowDebugMessage ( "\r\n测试其他命令...\r\n");
        //////////////////////////////////////
        //PXA_CMD27();
        //PXA_CMD55();
        PXA_ACMD6();

        //////////////////////////////////////
        PXA_ShowDebugMessage ( "\r\n goto SendCMD16 设置块大小...\r\n");

        goto SendCMD16;

SendCMD9://确认卡CSD.
//printf("+SendCMD9\r\n");
        WaitSpace();
        PXA_CMD9();

    //          displayCSDInfo((SDCSD*)mmcsdAttribT->mmc_csd);
        PXA_ShowDebugMessage ( "\r\n goto SendCMD7 选择确认地址的卡...\r\n");
        goto SendCMD7;

SendCMD16://设置块大小
    //printf("+SendCMD16\r\n" );
    WaitSpace();
    PXA_CMD16();
    PXA_ShowDebugMessage("\r\nSend CMD16..ok.\r\n");
    goto initend;
initend:
 goto SendAppCMD;

end:

    PXA_ShowDebugMessage("XllpMmcSdHwShutdown...\n");
 //XllpMmcSdHwShutdown(clkMgr);
 PXA_ExitSDMMC();
    PXA_ShowDebugMessage("XllpMmcSdHwShutdown...ok\n");
 WaitSpace();

 }


//////////////////////////////////////
// function prototypes
void PXA_Sleep1(DWORD    dwMicroSeconds)
{
 DWORD    startTime = OALGetTickCount();
    while ((OALGetTickCount() - startTime) < dwMicroSeconds)
    {
 }
}
```

```
//---------------------------------
///@brief 初始化系统的SDMMC卡
//---------------------------------
void PXA_InitSDMMC(P_XLLP_GPIO_T GpioReg, P_XLLP_MMC_T mmcRegs,
P_XLLP_CLKMGR_T ClckReg)
{
 g_pPXA_GpioReg = GpioReg;
 g_pPXA_ClckReg = ClckReg;
 g_pPXA_MmcReg = mmcRegs;
 XllpMmcSdHWInit(GpioReg,ClckReg);
 //PXA_Sleep1(20);
 //XllpMmcSetUpClock(g_pPXA_MmcReg, XLLP_MMC_304KHz, 0);//
 XllpMmcSetUpClock(g_pPXA_MmcReg, XLLP_MMC_19_5MHz, 0);//XLLP_MMC_19_5MHz
     g_pPXA_MmcReg->MMC_STRPCL = XLLP_STRPCL_STRTCLK;
 PXA_Sleep1(1);    //wait 1ms per MMC specification.
}

//---------------------------------
///@brief 卸载SDMMC卡
//---------------------------------
void PXA_ExitSDMMC()
{
 if(g_pPXA_ClckReg != NULL)
 {
     XllpMmcSdHwShutdown(g_pPXA_ClckReg);
 }
}

//-------------------------------------
///@brief 3.等待系统(等待时间=毫秒)
//-------------------------------------

void PXA_Sleep(DWORD dwMicroSeconds)
{
//   DWORD   startTime = OALGetTickCount();
//     while ((OALGetTickCount() - startTime) < dwMicroSeconds)
    {
 }
}



///@brief 4.显示调试信息
//#define PXA_ShowDebugMessage EdbgOutputDebugString
/*
void PXA_ShowDebugMessage(const unsigned char *sz, ...)
{
}
*/

//-------------------------------------
```

```
///@brief 5.检查状态
//---------------------------------------------
static UINT32 PXA_CheckStatus(P_XLLP_MMC_T mmcReg, UINT32 response)
{

 UINT32 StatusReceived=0;
 StatusReceived |= ((response & PXA_MMCSD_CARD_STATUS_CS)<<19);

    if(mmcReg->MMC_STAT & XLLP_STAT_TORD)
    StatusReceived |= PXA_MMCSD_STAT_TOR;

    if(mmcReg->MMC_STAT & XLLP_STAT_TORSPNS)
    StatusReceived |= PXA_MMCSD_STAT_TORSPN;

    if(mmcReg->MMC_STAT & XLLP_STAT_CRCWRERR)
    StatusReceived |= PXA_MMCSD_STAT_CRCWRER;

    if(mmcReg->MMC_STAT & XLLP_STAT_CRCRDERR)
    StatusReceived |= PXA_MMCSD_STAT_CRCRDERR;

    if(mmcReg->MMC_STAT & XLLP_STAT_DATAERRTKN)
    StatusReceived |= PXA_MMCSD_STAT_DATAERRTKN;

    if(mmcReg->MMC_STAT & XLLP_STAT_FLASHERR)
    StatusReceived |= PXA_MMCSD_STAT_FLASHERR;

    if(response & PXA_MMCSD_CARD_STATUS_OOR)
    StatusReceived |= PXA_MMCSD_STAT_OUT_OF_RANGE;

    if(response & PXA_MMCSD_CARD_STATUS_AE)
    StatusReceived |= PXA_MMCSD_STAT_ADDRESS_ERR;

    if(response & PXA_MMCSD_CARD_STATUS_EP )
    StatusReceived |= PXA_MMCSD_STAT_ERASE_PARAM;

    if(response & PXA_MMCSD_CARD_STATUS_WPV)
    StatusReceived |= PXA_MMCSD_CARD_STATUS_WPV;

    if(response & PXA_MMCSD_CARD_STATUS_CL)
    StatusReceived |= PXA_MMCSD_STAT_CARD_LOCKED;

    if(response & PXA_MMCSD_CARD_STATUS_ERR)
    StatusReceived |= PXA_MMCSD_STAT_ADDRESS_ERR;
    //////////////////////
    PXA_ShowDebugMessage("\r\nCMD MMC_STAT=0x%x  StatusReceived =
0x%x\r\n",mmcReg->MMC_STAT ,StatusReceived);

    return(StatusReceived);
 }

 //---------------------------------------------
 ///@brief 6. 分析返回状态
```

```
//---------------------------------
UINT32 PXA_ParseResponse( UINT16 * response, BOOL wrap)
{
   UINT32 tempResponse=0;

   if(wrap)
        tempResponse  =  ((((UINT32)response[2] & 0xff)<< 24) |
                         ((UINT32)response[1] << 8) | (response[0] & 0xff00)
>> 8 );
   else
     tempResponse  =  ((((UINT32)response[1] & 0xff)<< 24) |
                         ((UINT32)response[0] << 8) | (response[2] & 0xff00)
>> 8 );
   //////////////////////////////
  return(tempResponse);
}

//---------------------------------
///@brief 7.等待键盘输入
void PXA_WaitPushKey(char cKey )
{
    //while(0x20!=OEMReadDebugByte());
}

//--------------------------
///@brief 8.检查是否是SD卡
//--------------------------
BOOL PXA_IsSDCard()
{
 BOOL bSD = FALSE;
 PXA_STATRES statres;
 PXA_CMD55();
 statres = PXA_ACMD41();
 bSD = ((statres.status&XLLP_STAT_TORSPNS) != XLLP_STAT_TORSPNS);
 g_PXA_ISSDCARD = bSD ? 0x02 : 0x01;//SD = 2 MMC=1

 return bSD;
}

//------------------------------
///@brief 1.0 PXA_CMD_RESP 分析返回结果.供其他函数调用
//------------------------------
PXA_STATRES PXA_CMD_RESP()
{
    UINT16          res[8];
 PXA_STATRES     statusRes ;
 memset(&statusRes,0,sizeof(PXA_STATRES));
 //////////////////////////////////////////


 if(g_pPXA_MmcReg->MMC_CMDAT & XLLP_MMC_CMDAT_RESTYPE)
 {
```

```
        if((g_pPXA_MmcReg->MMC_CMDAT & 0x3) == XLLP_MMC_MMCSPIR2)
        {
        XllpMmcXtractResp(g_pPXA_MmcReg, res, XLLP_MMC_MMCSPIR2);
        PXA_ShowDebugMessage("\r\n...\r\n PXA res0=0x%x res1=0x%x res3=0x%x
res5=0x%x \r\n",res[0],res[1],res[3],res[5]);
        statusRes.lresponse[3] = PXA_ParseResponse(&res[5], 1);
        statusRes.lresponse[2] = PXA_ParseResponse(&res[3], 1);
        statusRes.lresponse[1] = PXA_ParseResponse(&res[1], 1);
        statusRes.lresponse[0] = PXA_ParseResponse(&res[0], 0);
        PXA_ShowDebugMessage("\r\n...\r\n Resp= 0x%x PXA res0=0x%x res1=0x%x
res2=0x%x\r\n",statusRes.sresponse,res[0],res[1],res[2]);
        PXA_ShowDebugMessage("\r\n...\r\n res3=0x%x res4=0x%x res5=0x%x
res6=0x%x res7=0x%x\r\n",res[3],res[4],res[5],res[6],res[7]);
        }
    else
    {
        XllpMmcXtractResp(g_pPXA_MmcReg, res, XLLP_MMC_MMCSPIR1);
        statusRes.sresponse = PXA_ParseResponse(&res[0], 1);
        PXA_ShowDebugMessage("\r\n...\r\n Resp= 0x%x PXA res0=0x%x res1=0x%x
res2=0x%x\r\n",statusRes.sresponse,res[0],res[1],res[2]);
        PXA_ShowDebugMessage("\r\n...\r\n res3=0x%x res4=0x%x res5=0x%x
res6=0x%x res7=0x%x\r\n",res[3],res[4],res[5],res[6],res[7]);
    }
 }
 ////////////////////////////////////
 statusRes.status = PXA_CheckStatus(g_pPXA_MmcReg,statusRes.sresponse);
 return statusRes;
 }

//-------------------------------------
///@brief 1.1 CMD0 .返回状态信息
//-------------------------------------
#define PXAMMC_WAITTIME 0x05    //等待时间

PXA_STATRES PXA_CMD0()
{
 PXA_STATRES statusRes ;
 memset(&statusRes,0,sizeof(PXA_STATRES));
 g_PXA_SDMMCOCR = 0x00;//OCR 寄存器

 PXA_ShowDebugMessage("\r\nSend CMD0...\r\n");

 XllpMmcSdInts(g_pPXA_MmcReg,XLLP_MMC_I_MASK_ENDCMDRES);
 XllpMmcSdSetupCmd(g_pPXA_MmcReg, XLLP_MMC_SPICMD0, 0, 64,0);
 //得到返回状态
   while (!(g_pPXA_MmcReg->MMC_STAT & XLLP_STAT_ENDCMDRES))
 {
   PXA_Sleep(PXAMMC_WAITTIME);
 }
```

```
    ////////////////////////////////////
    return PXA_CMD_RESP();
//  statusRes.status = PXA_CheckStatus(g_pPXA_MmcReg,statusRes.sresponse);
//  return statusRes;
}

//-------------------------------------
///@brief 1.2 CMD1 .返回状态信息
// XLLP_STAT_TORSPNS=超时=2
//-------------------------------------
PXA_STATRES PXA_CMD1()
{
 PXA_ShowDebugMessage("\r\nSend CMD1...\r\n");
 XllpMmcSdInts(g_pPXA_MmcReg,XLLP_MMC_I_MASK_ENDCMDRES);
 XllpMmcSdSetupCmd(g_pPXA_MmcReg, XLLP_MMC_SPICMD1, PXA_SDMMC_ADDR, 64,0);
 //得到返回状态
     while (!(g_pPXA_MmcReg->MMC_STAT & XLLP_STAT_ENDCMDRES))
 {
     PXA_Sleep(PXAMMC_WAITTIME);
 }
 return PXA_CMD_RESP();
}

//-------------------------------------
///@brief 1.2.1 ACMD41 .只有SD卡支持，MMC卡没有反映,在调用之间先调用CMD55
// XLLP_STAT_TORSPNS=超时=2
//-------------------------------------
PXA_STATRES PXA_ACMD41()
{
 PXA_ShowDebugMessage("\r\nSend ACMD41...\r\n");
 XllpMmcSdInts(g_pPXA_MmcReg,XLLP_MMC_I_MASK_ENDCMDRES);
 XllpMmcSdSetupCmd(g_pPXA_MmcReg, XLLP_SD_ACMD41, PXA_SDMMC_ADDR, 64,0);
 //得到返回状态
     while (!(g_pPXA_MmcReg->MMC_STAT & XLLP_STAT_ENDCMDRES))
 {
     PXA_Sleep(PXAMMC_WAITTIME);
 }
 return PXA_CMD_RESP();
}

//-------------------------------------
///@brief 1.2.2 CMD55 .只有SD卡支持，MMC卡没有反映
//-------------------------------------
PXA_STATRES PXA_CMD55()
{
 UINT32       SDMMC_ADDR = 0;
 PXA_ShowDebugMessage("\r\nSend CMD55...\r\n");
 XllpMmcSdInts(g_pPXA_MmcReg,XLLP_MMC_I_MASK_ENDCMDRES);
SDMMC_ADDR =  (g_PXA_SDMMCOCR & PXA_MMCSD_CARD_OCR_nBUSY) ? PXA_SDMMC_ADDR :
0x00;
 XllpMmcSdSetupCmd(g_pPXA_MmcReg, XLLP_MMC_SPICMD55, SDMMC_ADDR, 64,0);
 //得到返回状态
```

```
    while (!(g_pPXA_MmcReg->MMC_STAT & XLLP_STAT_ENDCMDRES))
  {
   PXA_Sleep(PXAMMC_WAITTIME);
  }
  return PXA_CMD_RESP();
  }

 //------------------------------------
 ///@brief 1.3 CMD2 .//ALL_SEND_CID
 //------------------------------------
 PXA_STATRES PXA_CMD2()
 {
  PXA_ShowDebugMessage("\r\nSend CMD2...\r\n");
  XllpMmcSdInts(g_pPXA_MmcReg,XLLP_MMC_I_MASK_ENDCMDRES);
  //得到所有的设备CID，发送的地址为0x0000
  XllpMmcSdSetupCmd(g_pPXA_MmcReg, XLLP_MMC_CMD2, 0x0000, 64,0);
  //得到返回状态
    while (!(g_pPXA_MmcReg->MMC_STAT & XLLP_STAT_ENDCMDRES))
  {
   PXA_Sleep(PXAMMC_WAITTIME);
  }
  return PXA_CMD_RESP();
  }

 //------------------------------------
 ///@brief 1.4 CMD3 .//SET_RELATIVE_ADDRESS
 //------------------------------------
 PXA_STATRES PXA_CMD3()
 {
  PXA_ShowDebugMessage("\r\nSend CMD3...\r\n");
  XllpMmcSdInts(g_pPXA_MmcReg,XLLP_MMC_I_MASK_ENDCMDRES);
  //关联地址，发送的地址为 PXA_SDMMC_ADDR
  if(g_PXA_ISSDCARD == 0x02)//SD卡
  {
       UINT16       res[8];
   PXA_STATRES    statusRes ;
   memset(&statusRes,0,sizeof(PXA_STATRES));
   XllpMmcSdSetupCmd(g_pPXA_MmcReg, XLLP_MMC_CMD3, 0x00, 64,0); //SD卡,
   //得到返回状态
   while (!(g_pPXA_MmcReg->MMC_STAT & XLLP_STAT_ENDCMDRES))
   {
       PXA_Sleep(PXAMMC_WAITTIME);
   }
   ////////////////////////////////////////////
  //  if(g_pPXA_MmcReg->MMC_CMDAT & XLLP_MMC_CMDAT_RESTYPE)
   {
       XllpMmcXtractResp(g_pPXA_MmcReg, res, XLLP_MMC_MMCSPIR1);
       statusRes.sresponse = PXA_ParseResponse(&res[0], 1);
       g_PXA_SD_ADDRESS = statusRes.sresponse & 0xFFFF0000;//得到SD卡重新分
配的地址
```

```
        }
        //////////////////////////////////
        statusRes.status = PXA_CheckStatus(g_pPXA_MmcReg,statusRes.sresponse);
        return statusRes;
    }
    else
    {
        XllpMmcSdSetupCmd(g_pPXA_MmcReg, XLLP_MMC_CMD3, PXA_SDMMC_ADDR, 64,0);
//MMC卡
    }
    //得到返回状态
       while (!(g_pPXA_MmcReg->MMC_STAT & XLLP_STAT_ENDCMDRES))
    {
        PXA_Sleep(PXAMMC_WAITTIME);
    }
    return PXA_CMD_RESP();
   }

   //------------------------------------
   ///@brief 1.5 CMD7 .//SELECT/DESELECT CARD
   //------------------------------------
   PXA_STATRES PXA_CMD7()
   {
    PXA_ShowDebugMessage("\r\nSend CMD7...\r\n");
    XllpMmcSdInts(g_pPXA_MmcReg,XLLP_MMC_I_MASK_ENDCMDRES);
    //关联地址，发送的地址为 PXA_SDMMC_ADDR
    if(g_PXA_ISSDCARD == 0x02)//SD卡
    {
       XllpMmcSdSetupCmd(g_pPXA_MmcReg, XLLP_MMC_CMD7, g_PXA_SD_ADDRESS, 64,0);
//SD
    }
    else
    {
        XllpMmcSdSetupCmd(g_pPXA_MmcReg, XLLP_MMC_CMD7, PXA_SDMMC_ADDR, 64,0);
    }
    //得到返回状态
       while (!(g_pPXA_MmcReg->MMC_STAT & XLLP_STAT_ENDCMDRES))
    {
        PXA_Sleep(PXAMMC_WAITTIME);
    }
    return PXA_CMD_RESP();
   }

   //------------------------------------
   ///@brief 1.6 CMD9 .//SEND_CSD
   //------------------------------------
   PXA_STATRES PXA_CMD9()
   {
    PXA_ShowDebugMessage("\r\nSend CMD9...\r\n");
    XllpMmcSdInts(g_pPXA_MmcReg,XLLP_MMC_I_MASK_ENDCMDRES);
    //关联地址，发送的地址为 PXA_SDMMC_ADDR
```

```
    if(g_PXA_ISSDCARD == 0x02)//SD卡
    {
        XllpMmcSdSetupCmd(g_pPXA_MmcReg, XLLP_MMC_SPICMD9, g_PXA_SD_ADDRESS,
64,0); //SD卡
    }
    else
    {
        XllpMmcSdSetupCmd(g_pPXA_MmcReg, XLLP_MMC_SPICMD9, PXA_SDMMC_ADDR,
64,0); //MMC卡
    }
    //得到返回状态
        while (!(g_pPXA_MmcReg->MMC_STAT & XLLP_STAT_ENDCMDRES))
    {
        PXA_Sleep(PXAMMC_WAITTIME);
    }
    return PXA_CMD_RESP();
    }


   //------------------------------------
   ///@brief 11.0 CMD27 .//PROGAM_CSD
   //------------------------------------
   PXA_STATRES  PXA_CMD27()
   {
    PXA_ShowDebugMessage("\r\nSend CMD27  PROGAM_CSD...\r\n");
    XllpMmcSdInts(g_pPXA_MmcReg,XLLP_MMC_I_MASK_ENDCMDRES);
    //
    XllpMmcSdSetupCmd(g_pPXA_MmcReg, XLLP_MMC_SPICMD27, 0x00, 64,0);

    //得到返回状态
        while (!(g_pPXA_MmcReg->MMC_STAT & XLLP_STAT_ENDCMDRES))
    {
        PXA_Sleep(PXAMMC_WAITTIME);
    }
    return PXA_CMD_RESP();
    }


   //------------------------------------
   ///@brief 11.1 ACMD6 .//SET_BUSWIDTH
   //------------------------------------
   PXA_STATRES  PXA_ACMD6()
   {
    PXA_ShowDebugMessage("\r\nSend ACMD6...\r\n");
    XllpMmcSdInts(g_pPXA_MmcReg,XLLP_MMC_I_MASK_ENDCMDRES);
    XllpMmcSdSetupCmd(g_pPXA_MmcReg, XLLP_SD_ACMD6, 0x00000002, 64,0);
    //得到返回状态
        while (!(g_pPXA_MmcReg->MMC_STAT & XLLP_STAT_ENDCMDRES))
    {
        PXA_Sleep(PXAMMC_WAITTIME);
    }
    return PXA_CMD_RESP();
```

```
}

//-----------------------------------
///@brief 1.7 CMD16 .//SET_BLOCKLEN
//-----------------------------------
PXA_STATRES PXA_CMD16()
{
 PXA_ShowDebugMessage("\r\nSend CMD16...\r\n");
 XllpMmcSdInts(g_pPXA_MmcReg,XLLP_MMC_I_MASK_ENDCMDRES);
 //参数定义，参数为块的大小=512字节，处理的是MMC卡
 XllpMmcSdSetupCmd(g_pPXA_MmcReg, XLLP_MMC_SPICMD16, PXA_BLOCK_SIZE, 64,0);
 //得到返回状态
    while (!(g_pPXA_MmcReg->MMC_STAT & XLLP_STAT_ENDCMDRES))
 {
     PXA_Sleep(PXAMMC_WAITTIME);
 }
 return PXA_CMD_RESP();
}

//-----------------------------------
///@brief 1.7 CMD17 .//READ_SINGLE_BLOCK
///@param dwSectorIndex = 扇区序号
///@param pBlockBuffer = 512字节的块缓冲区
//-----------------------------------
PXA_STATRES PXA_CMD17(DWORD dwSectorIndex,BYTE* pBlockBuffer)
{
 WORD    nReadDataCount = 0x00;//读的数据个数
 DWORD dwAddress = dwSectorIndex*PXA_BLOCK_SIZE;//得到数据开始地址

 //PXA_ShowDebugMessage("\r\nSend CMD17...\r\n");
 XllpMmcSdInts(g_pPXA_MmcReg,XLLP_MMC_I_MASK_ENDCMDRES);

 //只传送一块数据
    XllpMmcSdSetupXCmd(g_pPXA_MmcReg, PXA_BLOCK_SIZE, PXA_BLOCK_SIZE ,
0xffff);
 //查看第 dwSectorIndex 扇区
 XllpMmcSdSetupCmd(g_pPXA_MmcReg, XLLP_MMC_SPICMD17, dwAddress, 64,0);

 //设置读请求寄存器
 XllpMmcSdInts(g_pPXA_MmcReg, XLLP_MMC_I_MASK_RXFIFORDREQ);

 //等待数据传送完毕
 do
 {
     if((g_pPXA_MmcReg->MMC_I_REG & XLLP_MMC_I_REG_RXFIFORDREQ) ==
XLLP_MMC_I_REG_RXFIFORDREQ)
     {

         int i;
```

```
        //pBlockBuffer
        for( i = 0x00 ; i < 8 ; i++)
        {
            DWORD  dwRXData = g_pPXA_MmcReg->MB_RXFIFO;
            pBlockBuffer[nReadDataCount] = (BYTE)(dwRXData&0x000000FF);
            nReadDataCount++;
            pBlockBuffer[nReadDataCount] =
(BYTE)( (dwRXData&0x0000FF00)>>8 );
            nReadDataCount++;
            pBlockBuffer[nReadDataCount] =
(BYTE)( (dwRXData&0x00FF0000)>>16 );
            nReadDataCount++;
            pBlockBuffer[nReadDataCount] =
(BYTE)( (dwRXData&0xFF000000)>>24 );
            nReadDataCount++;
        }
    }

    }while ( (!(g_pPXA_MmcReg->MMC_STAT & XLLP_STAT_DATATRANDONE))||
(nReadDataCount< PXA_BLOCK_SIZE) );


  //得到返回状态
     while (!(g_pPXA_MmcReg->MMC_STAT & XLLP_STAT_ENDCMDRES))
  {
  //  PXA_Sleep(50);
  }
  //PXA_ShowDebugMessage("\r\n");
  //PXA_ShowDebugMessage("ReadDataCount=%d\r\n",nReadDataCount);
  //ShowBuffer(pBlockBuffer);


  {
    static  UINT16     cmd17res[8];
    static  PXA_STATRES     cmd17statusRes ;
    ////////////////////////////////////////////
    XllpMmcXtractResp(g_pPXA_MmcReg, cmd17res, XLLP_MMC_MMCSPIR1);
    return  cmd17statusRes;
  }

  return PXA_CMD_RESP();
 }

///////////////////////////////////////////////////
```

```
//FAT文件操作

///@brief 2.1 判断FAT类型,并返回根目录扇区序号和簇包含的扇区个数
/// (dwFirstRootDirSecIndex-nFATSecIndex)/2 就是FAT所占用的扇区个数,
///@param [in] pBPB
///@param [in/out] dwFirstRootDirSecIndex = 返回根目录扇区序号
///@param [in/out] nSectsPerClust = 返回每个文件簇包含的扇区个数
///@param [in/out] nFATSecIndex = 返回FAT扇区序号
///@return 返回FAT文件系统类型

PXA_FAT_TYPE CheckFATType(BIOSPB* pBPB, DWORD *dwFirstRootDirSecIndex, WORD*
nSectsPerClust, WORD* nFATSecIndex)
{
WORD     FATSz = 0x00;
WORD     TotSec = 0x00;
DWORD    DataSec = 0x00;
DWORD    RootDirSectors = 0x00;
DWORD    FirstDataSector = 0x00;
DWORD    CountofClusters = 0x00;
DWORD    FirstRootDirSecNum = 0x00;

//////////////////////////////////////
RootDirSectors = ( (pBPB->NumRootEntries*32) +
    (pBPB->BytesPerSect-1) )/pBPB->BytesPerSect;

if( pBPB->SectsPerFAT != 0)
{
    FATSz = pBPB->SectsPerFAT;//不是FAT32
}
if( pBPB->SectsPerPart != 0x00)
{
    TotSec = pBPB->SectsPerPart;//BPB_TotSec16
}
//FAT16文件系统根目录所在的扇区位置 FirstRootDirSecNum ，根目录占用32扇区
FirstRootDirSecNum = pBPB->RsvdSects + pBPB->NumFATs*FATSz;

FirstDataSector = pBPB->RsvdSects + pBPB->NumFATs*FATSz + RootDirSectors;//
开始扇区
DataSec = TotSec -
    (pBPB->RsvdSects + pBPB->NumFATs*FATSz + RootDirSectors);//总的数据扇区数

CountofClusters = DataSec/pBPB->SectsPerClust;//总文件簇
//////////////////////////////////////////////////
*nFATSecIndex = pBPB->RsvdSects;//FAT扇区序号
*dwFirstRootDirSecIndex = FirstRootDirSecNum;
*nSectsPerClust = pBPB->SectsPerClust;
g_DataSectorStart = FirstDataSector;//开始数据扇区
g_SectorsPerClust = pBPB->SectsPerClust;//簇包含扇区个数

if( CountofClusters < 4085 )
{
    // Volume is FAT12
```

```
        return PXA_FAT_TYPE_12;

    } else if( CountofClusters < 65525)
    {
        // Volume is FAT16
        return PXA_FAT_TYPE_16;

    } else
    {
        if(pBPB->SectsPerFAT != 0)
        {
            return PXA_FAT_TYPE_16;
        }
        //Volue is FAT32
        return PXA_FAT_TYPE_32;
    }
    return PXA_FAT_TYPE_NULL;

    /*
    CString str;
    str.Format("簇包含扇区数=%d FirstRootDirSecNum =%d FirstDataSector =%d
RootDirSectors=%d Clus = %d",

    pBPB->SectsPerClust,FirstRootDirSecNum,FirstDataSector,RootDirSectors,Coun
tofClusters);
    AfxMessageBox(str);

    GetRootDirFile(FirstRootDirSecNum,32,pBPB->SectsPerClust);

    */
   }
   ///@brief 2.2 初始化FAT数据
   ///@param pBlock=FAT数据
   ///@param PFT = FAT文件系统类型
   void InitFATData(BYTE* pBlock,PXA_FAT_TYPE PFT)
   {
    g_PXA_FAT_TYPE = PFT;
    memcpy(g_PXA_FAT,pBlock,512);
   }

   ///@brief 2.3 得到下一个文件簇
   ///@param nCurrentClust = 当前文件簇( >=2 )
   ///@return 返回下一个文件簇，如果是 0xFFFFFFFF 则没有下一个了
   DWORD   GetNextClust(DWORD nCurrentClust)
   {
    //现在只支持 FAT12/FAT16
    if( nCurrentClust<=1)   return nCurrentClust;
    if( PXA_FAT_TYPE_12==g_PXA_FAT_TYPE)
    {
        WORD    *pFAT = NULL;
        WORD    FATClust = 0x00;//当前簇指向的下一个文件簇序号
```

```
        WORD    nOffset = (WORD)nCurrentClust&0x0001;
        DWORD   FATOffset =  nCurrentClust +nOffset;
        if( nCurrentClust>=0x0FFF) return 0xFFFFFFFF;
        pFAT = (WORD*)&g_PXA_FAT[FATOffset];//最大512
        FATClust = pFAT[0];//
        FATClust = nOffset ? (FATClust&0x0FFF) : ( (FATClust>>4)&0x0FFFF ) ;
        return FATClust;
    }
    else if(PXA_FAT_TYPE_16==g_PXA_FAT_TYPE)
    {
        WORD    *pFAT = NULL;
        WORD    FATClust = 0x00;//当前簇指向的下一个文件簇序号
        if( nCurrentClust >= 0xFFFF)   return 0xFFFFFFFF;
        pFAT = (WORD*)&g_PXA_FAT[nCurrentClust*2];//最大512
        FATClust = pFAT[0];//
        return FATClust;
    }
    else
    {
        return 0xFFFFFFFF;
    }

}

///@brief 2.4 文件簇序号转换成扇区序号
///@param dwClusterIndex=文件簇序号
///@param ClusSecCount = 文件簇包含的扇区个数
///@return 扇区序号
DWORD   FileClusterToSector(DWORD dwClusterIndex, WORD ClusSecCount)
{
 //g_DataSectorStart = 0x00;//文件数据扇区开始位置。由得到FAT分区类型时得到
 DWORD ClusterOffset = dwClusterIndex - 2;//因为最开始的就是第2簇，所以要减掉2
 //具体扇区位置
 DWORD SecOffset = g_DataSectorStart + dwClusterIndex* ClusSecCount;//其中32
表示每簇包含的扇区数
 return SecOffset;
}


//FILEDIRECTORY;//文件目录结构
//得到根目录扇区的第一个扇区中的我们需要的文件(NK.BIN\NK.INI)目录结构
//static UINT32           g_NKFindFlags = 0x00;//是否找到对应的文件标志
//static FILEDIRECTORY    g_NK_INI_DIR;//NK  配置文件 NK.INI 目录信息
//static FILEDIRECTORY    g_NK_BIN_DIR;//NK  镜像文件 NK.BIN 目录信息

///@brief 2.5 计算根目录和对应的文件 FILEDIRECTORY
///@param    pDirSecBuffer = 根目录扇区缓冲区
void GetRootDirFile(BYTE* pDirSecBuffer)
{
 int nFileCount = 16;//每扇区可以用16个文件
 int j = 0x00;
```

```
    FILEDIRECTORY* pFile = (FILEDIRECTORY*)pDirSecBuffer;
    for( j = 0x00 ; j < nFileCount; j++)
    {
        if( pFile[j].FileAttr==0x0F ) continue;//长文件名，不支持
        if( pFile[j].FileAttr==0x10 ) continue;//目录，不支持
        if( pFile[j].FileAttr==0x08 ) continue;//卷标，不支持

        //已经删除的，不支持

    if((pFile[j].FileName[0]==0xE5)||(pFile[j].FileName[0]==0x00)||(pFile[j].F
ileName[0]==0x05))
        {
            continue;
        }
        ////////////////////////////////////
        if((pFile[j].FileName[0]==pFile[j].FileName[1])
            &&(pFile[j].FileName[0]==pFile[j].FileName[2])
            &&(pFile[j].FileName[0]==0x00))
        {
            goto SEARCH_FINISHED;//已经没有其他文件目录
        }
        //////////////////////////////////////////////
        //g_RootFile[g_RootFileCount];//找 NK.INI 文件

    if( (pFile[j].FileName[0]=='N')&&(pFile[j].FileName[1]=='K')&&(pFile[j].Fi
leName[2]==0x20) )
        {

    if( (pFile[j].ExtName[0]=='I')&&(pFile[j].ExtName[1]=='N')&&(pFile[j].ExtN
ame[2]=='I') )
            {
                memcpy(&g_NK_INI_DIR,&pFile[j],32);//找到了INI文件
                //FIND_NKBIN_FLAGS
                g_NKFindFlags |= FIND_NKINI_FLAGS;//
            }
        }

        //g_RootFile[g_RootFileCount];//找 NK.BIN 文件
    if( (pFile[j].FileName[0]=='N')&&(pFile[j].FileName[1]=='K')&&(pFile[j].Fi
leName[2]==0x20) )
        {

    if( (pFile[j].ExtName[0]=='B')&&(pFile[j].ExtName[1]=='I')&&(pFile[j].ExtN
ame[2]=='N') )
            {
                memcpy(&g_NK_BIN_DIR,&pFile[j],32);//找到了BIN文件
                //FIND_NKBIN_FLAGS
                g_NKFindFlags |= FIND_NKBIN_FLAGS;//
            }
        }
        //可以在此添加更多的文件信息查找
```

```
 }//end for j
 ///////////////////////////////////////////////
SEARCH_FINISHED:
 ;
 }

///@brief 2.6得到文件目录 FIND_NKBIN_DIR
///@param nFileIndex=指定的文件类型序号
///@return 返回具体的文件目录结构指针 FILEDIRECTORY
FILEDIRECTORY*  GetFileDir(DWORD nFileIndex)
{
 FILEDIRECTORY* pDir = NULL;
 if( (g_NKFindFlags&nFileIndex) != nFileIndex ) return NULL;
 /////////////////////////////////////////////////////////////
 switch(nFileIndex)
 {
 case FIND_NKBIN_FLAGS:
     pDir = &g_NK_BIN_DIR;
     break;
 case FIND_NKINI_FLAGS:
     pDir = &g_NK_INI_DIR;
     break;
 default:
     break;
 }
 return pDir;
}


///-----------------------------------------------------------------
///@brief 3. 读指定扇区的数据
///@param dwSectorIndex = 扇区开始序号
///@param nSecCount = 扇区个数
///@param pBlockBuffer = 块缓冲区地址
///-----------------------------------------------------------------
BOOL PXA_ReadSectors(DWORD dwSectorIndex,WORD nSecCount ,BYTE* pBlockBuffer)
{
 WORD i;
 PXA_STATRES status;
 for( i = 0x00 ; i < nSecCount ; i++)
 {
     status = PXA_CMD17(dwSectorIndex+i,&pBlockBuffer[i*512]);//读
 //  PXA_ShowDebugMessage("Info :  CMD17 status = 0x%x\r\n",status.status);

 }
 return TRUE;

}
```

实现 SD 卡读写的代码如下所示，对应的文件名为 PXA_SDMMC.C。

```
//-------------------------------------
//描述：PXA270 访问SD/MMC卡程序
//日期：2007-02-09
//修改记录：
//-------------------------------------
#include <windows.h>
#include <nkintr.h>
#include <bulverde.h>
#include <mainstoneii.h>
#include <oal_memory.h>
#include <pcireg.h>
#include <fmd.h>
#include <xllp_pccardsocket.h>
#include <bsp.h>
#include "loader.h"
#include <xllp_mmc.h>
#include "PXA_SDMMC.H"

///////////////////////////////////////
///@brief 外部引入函数定义
//extern msWait(unsigned msVal);
//extern int OEMReadDebugByte();
//extern void EdbgOutputDebugString( const unsigned char *sz, ...);
extern void ShowBuffer(BYTE* pBlock);
extern void  Launch(unsigned int uAddr);



///@brief 内部全局数据指针定义
static   P_XLLP_GPIO_T                  g_pPXA_GpioReg = NULL;
static   P_XLLP_CLKMGR_T                g_pPXA_ClckReg = NULL;
static   P_XLLP_MMC_T                   g_pPXA_MmcReg = NULL;
static   UINT32                         g_PXA_SDMMCOCR = 0x00;//OCR寄存器
static   WORD                           g_PXA_ISSDCARD = 0x00;//是否为SD卡 MMC= 1
SD = 2;
static   DWORD                          g_PXA_SD_ADDRESS = 0x00;//SD卡地址，因为SD
卡地址是通过CMD3返回的，这和MMC不同

///@brief 系统定义的SDMMC卡地址，而且系统只支持一个卡
//#define PXA_SDMMC_ADDR          0x200000
#define PXA_SDMMC_ADDR         0x420000
#define PXA_BLOCK_SIZE         0x000200     //扇区的大小=512字节


//------------------------------------------------------------------
//FILEDIRECTORY;//文件目录结构
static   UINT32          g_NKFindFlags = 0x00;//是否找到对应的文件标志
static   FILEDIRECTORY   g_NK_INI_DIR;//NK  配置文件 NK.INI 目录信息
static   FILEDIRECTORY   g_NK_BIN_DIR;//NK  镜像文件 NK.BIN 目录信息
```

```
static  DWORD        g_BiosReSectors = 0x00;//相对扇区数(Relative Sectors)
从该磁盘的开始到该分区的开始的位移量，以扇区来计算
static  DWORD        g_DataSectorStart = 0x00;//文件数据扇区开始位置。由得到
FAT分区类型时得到
static  WORD         g_SectorsPerClust = 0x00;//文件簇包含的扇区个数
static  BYTE         g_PXA_FAT[512] = { 0x00 };//文件分配表的第一个扇区内容
static  PXA_FAT_TYPE g_PXA_FAT_TYPE;//   FAT类型

#define FIND_NULL_FLAGS          0x00000000  //没有
#define FIND_NKBIN_FLAGS         0x00000001  //找到NK.BIN标志
#define FIND_NKINI_FLAGS         0x00000002  //找到NK.INI标志

//-----------------------------------------------------------------

typedef struct {
    UINT SCR_STRUCTURE : 4;
    UINT SD_SPEC : 4;
    UINT DATA_STAT_AFTER_ERASE : 1;
    UINT SD_SECURITY : 3;
    UINT SD_BUS_WIDTH : 4;
    UINT RES2 : 16;
    UINT RES1 : 32; // RESERVED FOR MANUFACTURER
}SCR;//SCR scr;

typedef struct
{
 UINT CMD;
 UINT ADDR;
 UINT FSIZE;
 UINT BSIZE;
 UINT RATE;
 UINT NOB;
 UINT DATATRANS;
 BOOL BUSY;
 BOOL READSCR;

}mmcStack;

typedef enum
{
  DNONE=0,
  WRITE,
  READ
}dataDirection;

typedef struct{
 UINT32 lresponse[4];
 UINT32 sresponse;
 UINT32 status;
}STATRES;
typedef struct {
    UINT SZ_PROC : 32;
```

```
        UINT SD_CARD_TYPE : 16;
        UINT RES2 : 13;
        UINT SECD_MODE : 1;
        UINT DAT_BUS_WIDTH : 2;
    } STATUS;
    typedef struct {
            UINT ECC : 2;                  // Bits 0-31
            UINT RES1 : 2;
            UINT TMP_WRITE_PROTECT : 1;
            UINT PERM_WRITE_PROTECT : 1;
            UINT COPY : 1;
            UINT RES2 : 1;
            UINT RES3 : 5;
            UINT WRITE_BL_PARTIAL : 1;
            UINT WRITE_BL_LEN : 4;
            UINT R2W_FACTOR : 3;
            UINT DEFAULT_ECC : 2;
            UINT WP_GRP_ENABLE : 1;
            UINT WP_GRP_SIZE : 5;
            UINT ERASE_GRP_MULT : 3;

            UINT ERASE_GRP_MULTx : 2;    // Bits 32-63
            UINT ERASE_GRP_SIZE : 5;
            UINT C_SIZE_MULT : 3;
            UINT VDD_W_CURR_MAX : 3;
            UINT VDD_W_CURR_MIN : 3;
            UINT VDD_R_CURR_MAX : 3;
            UINT VDD_R_CURR_MIN : 3;
            UINT C_SIZE : 10;

            UINT C_SIZEx : 2;             // Bits 64-95
            UINT RES4 : 2;
            UINT DSR_IMP : 1;
            UINT READ_BLK_MISALIGN : 1;
            UINT WRITE_BLK_MISALIGN : 1;
            UINT READ_BL_PARTIAL : 1;
            UINT READ_BL_LEN : 4;
            UINT CCC : 12;
            UINT TRAN_SPEED : 8;

            UINT NSAC : 8;                // Bits 96-127
            UINT TAAC : 8;
            UINT RES5 : 2;
            UINT SPEC_VERS : 4;
            UINT CSD_STRUCTURE : 2;
            UINT RES6 : 8;
    }MMCCSD;
    typedef struct {
        UINT NU : 1;     // 0- 31
        UINT CRC : 7;
        UINT MDT : 8;
        UINT PSN : 16;
```

```
    UINT PSNx : 16;  // 32- 63
    UINT PRV : 8;
    UINT PNM : 8;

    UINT PNMx :32;    // 64- 96

    UINT PNMxx : 7;   //96 - 127
    UINT OID : 16;
    UINT MID : 8;
}MMCCID;

typedef struct{
 UINT32 block;
 UINT32 *buffer;
 UINT size;
 BOOL mmc;
 UINT32 OCR;
 UINT32 rca;
 MMCCSD *mmc_csd;
 MMCCID *mmc_cid;
 SCR *sd_scr;
 STATUS sd_status;
}MMCSDAttribT;

#define SO_SCR          0x8;  // size of sd scr register
#define SO_BLOCK        0x200;
#define RCA_MMC_CARD    0x420000;
#define CMD1_ADDR       0x200000;

#define MMCSD_CARD_STATUS_OOR       (1u<<31)     // out of range
#define MMCSD_CARD_STATUS_AE        (1u<<30)    // address error
#define MMCSD_CARD_STATUS_EP        (1u<<27)    // erase param
#define MMCSD_CARD_STATUS_WPV       (1u<<26)    // write protected block
#define MMCSD_CARD_STATUS_CL        (1u<<25)    // card locked
#define MMCSD_CARD_STATUS_ERR       (1u<<19)    // general error
#define MMCSD_CARD_STATUS_CS        (0xfu<<9)   // current state
#define MMCSD_CARD_STATUS_RFD       (1u<<8)     // ready for data
#define MMCSD_CARD_STATUS_AC        (1u<<5)     // app command

#define MMCSD_CARD_OCR_nBUSY        (1u<<31)        //  app command

// this is the structure of the status response defined in the LLD

#define  MMCSD_STAT_TOR                    XLLP_STAT_TORD
#define  MMCSD_STAT_TORSPN                 XLLP_STAT_TORSPNS
#define  MMCSD_STAT_CRCWRER                XLLP_STAT_CRCWRERR
#define  MMCSD_STAT_CRCRDERR               XLLP_STAT_CRCRDERR

#define  MMCSD_STAT_DATAERRTKN             XLLP_STAT_DATAERRTKN
#define  MMCSD_STAT_RESCRCERR              XLLP_STAT_RESCRCERR
#define  MMCSD_STAT_FLASHERR               (1u<<6)
```

```
#define  MMCSD_STAT_CARD_ERR             (1u<<8)
#define  MMCSD_STAT_CARD_LOCKED          (1u<<9)
#define  MMCSD_STAT_WP_VIOLATION         (1u<<10)
#define  MMCSD_STAT_ERASE_PARAM          (1u<<11)
#define  MMCSD_STAT_ADDRESS_ERR          (1u<<12)
#define  MMCSD_STAT_OUT_OF_RANGE         (1u<<13)
#define  MMCSD_STAT_CARD_BUSY            (1u<<14)
#define  MMCSD_STAT_CURRENT_CARD_STATE   (0xfu<<28)


//----------------------------------------------------------------
/*
static void PXA_ShowDebugMessage(const unsigned char *sz, ...)
{
}

*/
///@brief 4.显示调试信息
#define PXA_ShowDebugMessage KITLOutputDebugString


void WaitSpace()
{
//   while(0x20!=OEMReadDebugByte());
}

///@brief 测试SDMMC卡
void TestSDMMC()
{
//   int index;
 int size=0;
 BOOL    bSD = FALSE;

 mmcStack          MMCSTK;
 STATRES           mmcsdStatus;
 MMCSDAttribT      mmcsdAttribT;
//   MMCCID            mmccid;
//   MMCCSD            mcccsd;
 XLLP_CLKMGR_T    *clkMgr = NULL;//(XLLP_CLKMGR_T*) CLK_BASE_U_VIRTUAL;
 XLLP_GPIO_T      *pGPIO = NULL;
 XLLP_MMC_T       *pMMC = NULL;//(XLLP_MMC_T* ) MMC_BASE_U_VIRTUAL;
    PXA_ShowDebugMessage ( "\r\n 测试SDMMC卡 ...\r\n");

 pGPIO  = (XLLP_GPIO_T *) OALPAtoVA(BULVERDE_BASE_REG_PA_GPIO,
FALSE);//FALSE=U TUE=V
 pMMC   = (XLLP_MMC_T *) OALPAtoVA(BULVERDE_BASE_REG_PA_MMC, FALSE);//FALSE=U
TUE=V
 clkMgr = (XLLP_CLKMGR_T *) OALPAtoVA(BULVERDE_BASE_REG_PA_CLKMGR,
FALSE);//FALSE=U TUE=V
```

```
PXA_InitSDMMC(pGPIO,pMMC,clkMgr);

PXA_CMD0();

if(PXA_IsSDCard())
{
        PXA_ShowDebugMessage ( "\r\n 是SD卡...\r\n");
    bSD = TRUE;
    PXA_CMD55();
    {
        PXA_STATRES temp =  PXA_ACMD41();
        mmcsdStatus = *((STATRES*)&temp);
    }
    mmcsdAttribT.OCR  = mmcsdStatus.sresponse;
}
else
{
    PXA_STATRES temp = PXA_CMD1();
    mmcsdStatus = *((STATRES*)&temp);
    if( mmcsdStatus.status == XLLP_STAT_TORSPNS)//超时
    {
        PXA_ShowDebugMessage ( "\r\n 没有卡 ...\r\n");
        WaitSpace();
        return ;
    }
        PXA_ShowDebugMessage ( "\r\n 发现了MMC卡 ...\r\n");
    mmcsdAttribT.OCR  = mmcsdStatus.sresponse;
}


goto SendCMD2_1;
///////////////////////////////////


SendAppCMD:
{
    DWORD   MBR_OFFSET = 0x00;//主引导扇区保留
    DWORD   dwFRSI = 0x00;
    MBR_DPT*    pDPT = NULL;
    BYTE buffer[4096];
    pDPT = (MBR_DPT*)&buffer[0x01BE];
    PXA_CMD17(0,buffer);//读MBR
    ShowBuffer(buffer);
    PXA_ShowDebugMessage("BootID=0x%x SID=0x%x RS=0x%x TS=0x%x\r\n",
          pDPT->BootID,pDPT->SystemID,pDPT->ReSectors,pDPT->TotalSec);

    ///////////////////////////////////////
    PXA_ShowDebugMessage("读 BIOSPB 扇区\r\n");
    WaitSpace();
    MBR_OFFSET = pDPT->ReSectors;
    PXA_CMD17(pDPT->ReSectors, buffer);
```

```
        ShowBuffer(buffer);
        //BIOSPB PXA_FAT_TYPE CheckFATType(BIOSPB* pBPB, DWORD
*dwFirstRootDirSecIndex, WORD* nSectsPerClust, WORD* nFATSecIndex)
        {
            PXA_FAT_TYPE    PFT;
            DWORD    dwFirstRootDirSecIndex;
            WORD     nSectsPerClust;
            WORD     nFATSecIndex;
            PFT =
CheckFATType((BIOSPB*)buffer,&dwFirstRootDirSecIndex,&nSectsPerClust,&nFATSecI
ndex);
            PXA_ShowDebugMessage("PFT= %d FRSI=0x%x   SPC=0x%x
FSI=0x%x\r\n",
                PFT,dwFirstRootDirSecIndex,nSectsPerClust,nFATSecIndex);
            ///////////////////////////////////////////////////////////////
            dwFRSI = dwFirstRootDirSecIndex;
        }
        WaitSpace();

        //初始化文件系统

        PXA_InitFAT(buffer,(WORD)MBR_OFFSET);
        {
            UCHAR* strFileName = "NK.BIN";
            DWORD    dwFileSize = 0x00;
    //          BYTE    MYBuf[0x200];
            WORD     readSize = 0x200;
            DWORD    ReadCount = 0x00;
            DWORD i = 0;

            //
            // Open the BIN file for reading
            //
            {
                ULONG ImageLoc = 0x00;
                ULONG ImagePhLoc = 0x00;
                if(!PXA_LoadBINFile(strFileName , &ImageLoc ))
                {
                    PXA_ShowDebugMessage("ERROR: LoadBinFile Failed .\r\n");
                }
                else
                {
                    ImagePhLoc = (UINT32) OALVAtoPA((VOID *)ImageLoc);
                    PXA_ShowDebugMessage("INFO: LoadBinFile
OK .VStartAddress=0x%x \r\n",ImageLoc);
                    PXA_ShowDebugMessage("INFO: LoadBinFile
OK .PStartAddress=0x%x \r\n",ImagePhLoc);

                }
                ///////////////
```

```
                PXA_ExitSDMMC();
                PXA_ShowDebugMessage("INFO: Push K Key to Launch CE...\r\n");
    /*
                while(1)
                {
                    int key =OEMReadDebugByte();
                    if( (key == 'K')||( key == 'k') )
                    {
                        break;
                    }

                }
    */
                WaitSpace();
                Launch(ImagePhLoc);

            }
            //PXA_CloseFile();

        }

        return;


        {
            WORD DirOffset = (WORD)(MBR_OFFSET + dwFRSI);
            PXA_ShowDebugMessage("读主目录扇区 0x%x \r\n",DirOffset);
            PXA_CMD17( DirOffset, buffer);
            ShowBuffer(buffer);
            //void  GetRootDirFile(BYTE* pDirSecBuffer)//得到文件目录
            GetRootDirFile(buffer);
            {
                //FIND_NKINI_DIR
                FILEDIRECTORY* pDIRINI = GetFileDir(FIND_NKINI_DIR);
                FILEDIRECTORY* pDIRBIN = GetFileDir(FIND_NKBIN_DIR);
                if( pDIRINI != NULL)
                {
                    PXA_ShowDebugMessage("找到了文件 NK.INI\r\n");
                    PXA_ShowDebugMessage("开始簇号=0x%x 文件长度
=%d\r\n",pDIRINI->CulsterLow,pDIRINI->FileLength);
                }
                if( pDIRBIN != NULL)
                {
                    int j = 0;
                    DWORD i;

                    PXA_ShowDebugMessage("找到了文件 NK.BIN\r\n");
```

```
                    PXA_ShowDebugMessage("开始簇号=0x%x 文件长度
=%d\r\n",pDIRBIN->CulsterLow,pDIRBIN->FileLength);
                    PXA_ShowDebugMessage("\n....按空格键开始读取32M数据...\r\n");
                    WaitSpace();

                    for( i = 0 ; i < 2048*32 ; i++)
                    {
                        PXA_CMD17( DirOffset, buffer);
                    }
                    PXA_ShowDebugMessage("read Data %d M\r\n",32);


                //  ShowBuffer(buffer);

                }
            }
        }




    }
    goto end;

    SendCMD1:
    {
        PXA_STATRES temp ;
        if( !bSD)
        {
            temp = PXA_CMD1();
        }
        else
        {//SD卡
            PXA_CMD55();
            temp = PXA_ACMD41();
        }
        mmcsdStatus = *((STATRES*)&temp);
        mmcsdAttribT.OCR  = mmcsdStatus.sresponse;
        if (mmcsdAttribT.OCR & MMCSD_CARD_OCR_nBUSY)  // Card power up status
bit (!busy)
        {

            mmcsdAttribT.mmc = bSD ? XLLP_FALSE : XLLP_TRUE;
                MMCSTK.BUSY = XLLP_FALSE;
            PXA_ShowDebugMessage ( "\r\n go to SendCMD2 .广播,得所有卡CID..\r\n");
                goto SendCMD2;
            }
            else
            {
```

```
                PXA_ShowDebugMessage ( "\r\n next go to SendCMD1 ...\r\n");
                goto SendCMD1;
            }
    }

    SendCMD2_1:
    if (!(mmcsdStatus.sresponse & MMCSD_CARD_OCR_nBUSY))  // Card power up
status bit (!busy)
        {
        PXA_ShowDebugMessage ( "\r\n goto SendCMD1...\r\n");
        goto SendCMD1;
        }
        else
        {
            MMCSTK.BUSY = XLLP_FALSE;
            PXA_ShowDebugMessage ( "\r\n SendCMD2_1 go to SendCMD2...\r\n");
            goto SendCMD2;
        }

    SendCMD2://发送广播命令，得到所有卡的CID
    //printf("+SendCMD2\r\n");
        WaitSpace();
        PXA_CMD2();

            PXA_ShowDebugMessage ( "\r\n goto SendCMD3 确认卡地址...\r\n");
            goto SendCMD3;

    SendCMD3://确认卡地址
    //printf("+SendCMD3\r\n");
        {
            WaitSpace();
            PXA_CMD3();
            goto EndInitClock;
        }

    EndInitClock:
    //printf("+EndInitClock\r\n");
    //              XllpMmcSdInts(MmcReg,XLLP_MMC_I_MASK_CLKISOFF);
    //          MmcReg->MMC_STRPCL = XLLP_STRPCL_STPCLK;          // Stop the MMC
clock
    //              osStatus = OSAFlagWait(_keypadFlagRef, KEYPAD_FLAG_MASK,
OSA_FLAG_OR_CLEAR, &keypadFlags, OSA_SUSPEND);
    //          ASSERT(osStatus == OS_SUCCESS);
    //              while (MmcReg->MMC_STAT & XLLP_STAT_CLKEN);    // Wait for
the clock to stop
    //          XllpMmcSetUpClock(MmcReg, XLLP_MMC_19_5MHz, 1);
            PXA_ShowDebugMessage ( "\r\n goto SendCMD9 确认卡CSD...\r\n");
            goto SendCMD9;

    SendCMD7://选择确认地址的卡
            WaitSpace();
```

```
        PXA_CMD7();
        ///////////////////
        PXA_ShowDebugMessage ( "\r\n测试其他命令...\r\n");
        //////////////////////////////////////
        //PXA_CMD27();
        //PXA_CMD55();
        PXA_ACMD6();

        //////////////////////////////////////
        PXA_ShowDebugMessage ( "\r\n goto SendCMD16 设置块大小...\r\n");

        goto SendCMD16;

SendCMD9://确认卡CSD.
//printf("+SendCMD9\r\n");
        WaitSpace();
        PXA_CMD9();

    //          displayCSDInfo((SDCSD*)mmcsdAttribT->mmc_csd);
        PXA_ShowDebugMessage ( "\r\n goto SendCMD7 选择确认地址的卡...\r\n");
        goto SendCMD7;

SendCMD16://设置块大小
    //printf("+SendCMD16\r\n" );
    WaitSpace();
    PXA_CMD16();
    PXA_ShowDebugMessage("\r\nSend CMD16..ok.\r\n");
    goto initend;
initend:
 goto SendAppCMD;

end:

    PXA_ShowDebugMessage("XllpMmcSdHwShutdown...\n");
 //XllpMmcSdHwShutdown(clkMgr);
 PXA_ExitSDMMC();
    PXA_ShowDebugMessage("XllpMmcSdHwShutdown...ok\n");
 WaitSpace();

}


///////////////////////////////////////
// function prototypes

void PXA_Sleep1(DWORD    dwMicroSeconds)
{
 DWORD   startTime = OALGetTickCount();
    while ((OALGetTickCount() - startTime) < dwMicroSeconds)
    {
 }
}
```

```
//--------------------------------
///@brief 初始化系统的SDMMC卡
//--------------------------------
void PXA_InitSDMMC(P_XLLP_GPIO_T GpioReg, P_XLLP_MMC_T mmcRegs,
P_XLLP_CLKMGR_T ClckReg)
{
 g_pPXA_GpioReg = GpioReg;
 g_pPXA_ClckReg = ClckReg;
 g_pPXA_MmcReg = mmcRegs;
 XllpMmcSdHWInit(GpioReg,ClckReg);
 //PXA_Sleep1(20);
 //XllpMmcSetUpClock(g_pPXA_MmcReg, XLLP_MMC_304KHz, 0);//
 XllpMmcSetUpClock(g_pPXA_MmcReg, XLLP_MMC_19_5MHz, 0);//XLLP_MMC_19_5MHz
     g_pPXA_MmcReg->MMC_STRPCL = XLLP_STRPCL_STRTCLK;
 PXA_Sleep1(1);    //wait 1ms per MMC specification.
}

//-----------------------------
///@brief 卸载SDMMC卡
//-----------------------------
void PXA_ExitSDMMC()
{
 if(g_pPXA_ClckReg != NULL)
 {
     XllpMmcSdHwShutdown(g_pPXA_ClckReg);
 }
}

//-----------------------------------
///@brief 3.等待系统(等待时间=毫秒)
//-----------------------------------

void PXA_Sleep(DWORD dwMicroSeconds)
{
//   DWORD   startTime = OALGetTickCount();
//     while ((OALGetTickCount() - startTime) < dwMicroSeconds)
    {
 }
}


///@brief 4.显示调试信息
//#define PXA_ShowDebugMessage EdbgOutputDebugString
/*
void PXA_ShowDebugMessage(const unsigned char *sz, ...)
{
}
*/

//-----------------------------------
```

```
///@brief 5.检查状态
//------------------------------------------
static UINT32 PXA_CheckStatus(P_XLLP_MMC_T mmcReg, UINT32 response)
{

 UINT32 StatusReceived=0;
 StatusReceived |= ( (response & PXA_MMCSD_CARD_STATUS_CS)<<19);

    if(mmcReg->MMC_STAT & XLLP_STAT_TORD)
    StatusReceived |= PXA_MMCSD_STAT_TOR;

    if(mmcReg->MMC_STAT & XLLP_STAT_TORSPNS)
    StatusReceived |= PXA_MMCSD_STAT_TORSPN;

    if(mmcReg->MMC_STAT & XLLP_STAT_CRCWRERR)
    StatusReceived |= PXA_MMCSD_STAT_CRCWRER;

    if(mmcReg->MMC_STAT & XLLP_STAT_CRCRDERR)
    StatusReceived |= PXA_MMCSD_STAT_CRCRDERR;

    if(mmcReg->MMC_STAT & XLLP_STAT_DATAERRTKN)
    StatusReceived |= PXA_MMCSD_STAT_DATAERRTKN;

    if(mmcReg->MMC_STAT & XLLP_STAT_FLASHERR)
    StatusReceived |= PXA_MMCSD_STAT_FLASHERR;

    if(response & PXA_MMCSD_CARD_STATUS_OOR)
    StatusReceived |= PXA_MMCSD_STAT_OUT_OF_RANGE;

    if(response & PXA_MMCSD_CARD_STATUS_AE)
    StatusReceived |= PXA_MMCSD_STAT_ADDRESS_ERR;

    if(response & PXA_MMCSD_CARD_STATUS_EP )
    StatusReceived |= PXA_MMCSD_STAT_ERASE_PARAM;

    if(response & PXA_MMCSD_CARD_STATUS_WPV)
    StatusReceived |= PXA_MMCSD_CARD_STATUS_WPV;

    if(response & PXA_MMCSD_CARD_STATUS_CL)
    StatusReceived |= PXA_MMCSD_STAT_CARD_LOCKED;

    if(response & PXA_MMCSD_CARD_STATUS_ERR)
    StatusReceived |= PXA_MMCSD_STAT_ADDRESS_ERR;
    //////////////////////
    PXA_ShowDebugMessage("\r\nCMD MMC_STAT=0x%x  StatusReceived =
0x%x\r\n",mmcReg->MMC_STAT ,StatusReceived);

    return(StatusReceived);
}

//------------------------------------------
///@brief 6. 分析返回状态
```

```
//--------------------------------
UINT32 PXA_ParseResponse( UINT16 * response, BOOL wrap)
{
  UINT32 tempResponse=0;

  if(wrap)
       tempResponse = ((((UINT32)response[2] & 0xff)<< 24) |
                    ((UINT32)response[1] << 8) | (response[0] & 0xff00)
>> 8 );
  else
    tempResponse  =  ((((UINT32)response[1] & 0xff)<< 24) |
                    ((UINT32)response[0] << 8) | (response[2] & 0xff00)
>> 8 );
  //////////////////////////////
 return(tempResponse);
}

//--------------------------------
///@brief 7.等待键盘输入
void PXA_WaitPushKey(char cKey )
{
   //while(0x20!=OEMReadDebugByte());
}

//---------------------------
///@brief 8.检查是否是SD卡
//---------------------------
BOOL PXA_IsSDCard()
{
 BOOL bSD = FALSE;
 PXA_STATRES statres;
 PXA_CMD55();
 statres = PXA_ACMD41();
 bSD = ((statres.status&XLLP_STAT_TORSPNS) != XLLP_STAT_TORSPNS);
 g_PXA_ISSDCARD = bSD ? 0x02 : 0x01;//SD = 2 MMC=1

 return bSD;
}

//--------------------------------
///@brief 1.0 PXA_CMD_RESP 分析返回结果.供其他函数调用
//--------------------------------
PXA_STATRES PXA_CMD_RESP()
{
   UINT16          res[8];
 PXA_STATRES     statusRes ;
 memset(&statusRes,0,sizeof(PXA_STATRES));
 ////////////////////////////////////////////

 if(g_pPXA_MmcReg->MMC_CMDAT & XLLP_MMC_CMDAT_RESTYPE)
 {
```

```
        if((g_pPXA_MmcReg->MMC_CMDAT & 0x3) == XLLP_MMC_MMCSPIR2)
        {
        XllpMmcXtractResp(g_pPXA_MmcReg, res, XLLP_MMC_MMCSPIR2);
        PXA_ShowDebugMessage("\r\n...\r\n PXA res0=0x%x res1=0x%x res3=0x%x
res5=0x%x \r\n",res[0],res[1],res[3],res[5]);
        statusRes.lresponse[3] = PXA_ParseResponse(&res[5], 1);
        statusRes.lresponse[2] = PXA_ParseResponse(&res[3], 1);
        statusRes.lresponse[1] = PXA_ParseResponse(&res[1], 1);
        statusRes.lresponse[0] = PXA_ParseResponse(&res[0], 0);
        PXA_ShowDebugMessage("\r\n...\r\n Resp= 0x%x PXA res0=0x%x res1=0x%x
res2=0x%x\r\n",statusRes.sresponse,res[0],res[1],res[2]);
        PXA_ShowDebugMessage("\r\n...\r\n res3=0x%x res4=0x%x res5=0x%x
res6=0x%x res7=0x%x\r\n",res[3],res[4],res[5],res[6],res[7]);
        }
     else
     {
        XllpMmcXtractResp(g_pPXA_MmcReg, res, XLLP_MMC_MMCSPIR1);
        statusRes.sresponse = PXA_ParseResponse(&res[0], 1);
        PXA_ShowDebugMessage("\r\n...\r\n Resp= 0x%x PXA res0=0x%x res1=0x%x
res2=0x%x\r\n",statusRes.sresponse,res[0],res[1],res[2]);
        PXA_ShowDebugMessage("\r\n...\r\n res3=0x%x res4=0x%x res5=0x%x
res6=0x%x res7=0x%x\r\n",res[3],res[4],res[5],res[6],res[7]);
     }
  }
  ////////////////////////////////////
  statusRes.status = PXA_CheckStatus(g_pPXA_MmcReg,statusRes.sresponse);
  return statusRes;
  }

  //-----------------------------------
  ///@brief 1.1 CMD0 .返回状态信息
  //-----------------------------------
  #define PXAMMC_WAITTIME 0x05    //等待时间

  PXA_STATRES PXA_CMD0()
  {
  PXA_STATRES statusRes ;
  memset(&statusRes,0,sizeof(PXA_STATRES));
  g_PXA_SDMMCOCR = 0x00;//OCR 寄存器

  PXA_ShowDebugMessage("\r\nSend CMD0...\r\n");

  XllpMmcSdInts(g_pPXA_MmcReg,XLLP_MMC_I_MASK_ENDCMDRES);
  XllpMmcSdSetupCmd(g_pPXA_MmcReg, XLLP_MMC_SPICMD0, 0, 64,0);
  //得到返回状态
     while (!(g_pPXA_MmcReg->MMC_STAT & XLLP_STAT_ENDCMDRES))
  {
    PXA_Sleep(PXAMMC_WAITTIME);
  }
```

```
/////////////////////////////////////
return PXA_CMD_RESP();
//  statusRes.status = PXA_CheckStatus(g_pPXA_MmcReg,statusRes.sresponse);
//  return statusRes;
}

//-----------------------------------
///@brief 1.2 CMD1 .返回状态信息
// XLLP_STAT_TORSPNS=超时=2
//-----------------------------------
PXA_STATRES PXA_CMD1()
{
 PXA_ShowDebugMessage("\r\nSend CMD1...\r\n");
 XllpMmcSdInts(g_pPXA_MmcReg,XLLP_MMC_I_MASK_ENDCMDRES);
 XllpMmcSdSetupCmd(g_pPXA_MmcReg, XLLP_MMC_SPICMD1, PXA_SDMMC_ADDR, 64,0);
 //得到返回状态
    while (!(g_pPXA_MmcReg->MMC_STAT & XLLP_STAT_ENDCMDRES))
 {
    PXA_Sleep(PXAMMC_WAITTIME);
 }
 return PXA_CMD_RESP();
}

//-----------------------------------
///@brief 1.2.1 ACMD41 .只有SD卡支持，MMC卡没有反映,在调用之间先调用CMD55
// XLLP_STAT_TORSPNS=超时=2
//-----------------------------------
PXA_STATRES PXA_ACMD41()
{
 PXA_ShowDebugMessage("\r\nSend ACMD41...\r\n");
 XllpMmcSdInts(g_pPXA_MmcReg,XLLP_MMC_I_MASK_ENDCMDRES);
 XllpMmcSdSetupCmd(g_pPXA_MmcReg, XLLP_SD_ACMD41, PXA_SDMMC_ADDR, 64,0);
 //得到返回状态
    while (!(g_pPXA_MmcReg->MMC_STAT & XLLP_STAT_ENDCMDRES))
 {
    PXA_Sleep(PXAMMC_WAITTIME);
 }
 return PXA_CMD_RESP();
}

//-----------------------------------
///@brief 1.2.2 CMD55 .只有SD卡支持，MMC卡没有反映
//-----------------------------------
PXA_STATRES PXA_CMD55()
{
 UINT32      SDMMC_ADDR = 0;
 PXA_ShowDebugMessage("\r\nSend CMD55...\r\n");
 XllpMmcSdInts(g_pPXA_MmcReg,XLLP_MMC_I_MASK_ENDCMDRES);
SDMMC_ADDR = (g_PXA_SDMMCOCR & PXA_MMCSD_CARD_OCR_nBUSY) ? PXA_SDMMC_ADDR :
0x00;
 XllpMmcSdSetupCmd(g_pPXA_MmcReg, XLLP_MMC_SPICMD55, SDMMC_ADDR, 64,0);
 //得到返回状态
```

```
    while (!(g_pPXA_MmcReg->MMC_STAT & XLLP_STAT_ENDCMDRES))
 {
    PXA_Sleep(PXAMMC_WAITTIME);
 }
 return PXA_CMD_RESP();
}

//-----------------------------------
///@brief 1.3 CMD2 .//ALL_SEND_CID
//-----------------------------------
PXA_STATRES PXA_CMD2()
{
 PXA_ShowDebugMessage("\r\nSend CMD2...\r\n");
 XllpMmcSdInts(g_pPXA_MmcReg,XLLP_MMC_I_MASK_ENDCMDRES);
 //得到所有的设备CID，发送的地址为0x0000
 XllpMmcSdSetupCmd(g_pPXA_MmcReg, XLLP_MMC_CMD2, 0x0000, 64,0);
 //得到返回状态
    while (!(g_pPXA_MmcReg->MMC_STAT & XLLP_STAT_ENDCMDRES))
 {
    PXA_Sleep(PXAMMC_WAITTIME);
 }
 return PXA_CMD_RESP();
}


//-----------------------------------
///@brief 1.4 CMD3 .//SET_RELATIVE_ADDRESS
//-----------------------------------
PXA_STATRES PXA_CMD3()
{
 PXA_ShowDebugMessage("\r\nSend CMD3...\r\n");
 XllpMmcSdInts(g_pPXA_MmcReg,XLLP_MMC_I_MASK_ENDCMDRES);
 //关联地址，发送的地址为 PXA_SDMMC_ADDR
 if(g_PXA_ISSDCARD == 0x02)//SD卡
 {
        UINT16          res[8];
    PXA_STATRES     statusRes ;
    memset(&statusRes,0,sizeof(PXA_STATRES));
    XllpMmcSdSetupCmd(g_pPXA_MmcReg, XLLP_MMC_CMD3, 0x00, 64,0); //SD卡,
    //得到返回状态
    while (!(g_pPXA_MmcReg->MMC_STAT & XLLP_STAT_ENDCMDRES))
    {
        PXA_Sleep(PXAMMC_WAITTIME);
    }
    ////////////////////////////////////////////
 //  if(g_pPXA_MmcReg->MMC_CMDAT & XLLP_MMC_CMDAT_RESTYPE)
    {
        XllpMmcXtractResp(g_pPXA_MmcReg, res, XLLP_MMC_MMCSPIR1);
        statusRes.sresponse = PXA_ParseResponse(&res[0], 1);
        g_PXA_SD_ADDRESS = statusRes.sresponse & 0xFFFF0000;//得到SD卡重新分
配的地址
```

```
        }
        //////////////////////////////////
        statusRes.status = PXA_CheckStatus(g_pPXA_MmcReg,statusRes.sresponse);
        return statusRes;
    }
    else
    {
        XllpMmcSdSetupCmd(g_pPXA_MmcReg, XLLP_MMC_CMD3, PXA_SDMMC_ADDR, 64,0);
//MMC卡
    }
    //得到返回状态
       while (!(g_pPXA_MmcReg->MMC_STAT & XLLP_STAT_ENDCMDRES))
    {
        PXA_Sleep(PXAMMC_WAITTIME);
    }
    return PXA_CMD_RESP();
    }

    //------------------------------------
    ///@brief 1.5 CMD7 .//SELECT/DESELECT CARD
    //------------------------------------
    PXA_STATRES PXA_CMD7()
    {
    PXA_ShowDebugMessage("\r\nSend CMD7...\r\n");
    XllpMmcSdInts(g_pPXA_MmcReg,XLLP_MMC_I_MASK_ENDCMDRES);
    //关联地址，发送的地址为 PXA_SDMMC_ADDR
    if(g_PXA_ISSDCARD == 0x02)//SD卡
    {
       XllpMmcSdSetupCmd(g_pPXA_MmcReg, XLLP_MMC_CMD7, g_PXA_SD_ADDRESS, 64,0);
//SD
    }
    else
    {
        XllpMmcSdSetupCmd(g_pPXA_MmcReg, XLLP_MMC_CMD7, PXA_SDMMC_ADDR, 64,0);
    }
    //得到返回状态
       while (!(g_pPXA_MmcReg->MMC_STAT & XLLP_STAT_ENDCMDRES))
    {
        PXA_Sleep(PXAMMC_WAITTIME);
    }
    return PXA_CMD_RESP();
    }

    //------------------------------------
    ///@brief 1.6 CMD9 .//SEND_CSD
    //------------------------------------
    PXA_STATRES PXA_CMD9()
    {
    PXA_ShowDebugMessage("\r\nSend CMD9...\r\n");
    XllpMmcSdInts(g_pPXA_MmcReg,XLLP_MMC_I_MASK_ENDCMDRES);
    //关联地址，发送的地址为 PXA_SDMMC_ADDR
```

```
    if(g_PXA_ISSDCARD == 0x02)//SD卡
    {
        XllpMmcSdSetupCmd(g_pPXA_MmcReg, XLLP_MMC_SPICMD9, g_PXA_SD_ADDRESS,
64,0); //SD卡
    }
    else
    {
        XllpMmcSdSetupCmd(g_pPXA_MmcReg, XLLP_MMC_SPICMD9, PXA_SDMMC_ADDR,
64,0); //MMC卡
    }
    //得到返回状态
       while (!(g_pPXA_MmcReg->MMC_STAT & XLLP_STAT_ENDCMDRES))
    {
        PXA_Sleep(PXAMMC_WAITTIME);
    }
    return PXA_CMD_RESP();
   }


   //------------------------------------
   ///@brief 11.0 CMD27 .//PROGAM_CSD
   //------------------------------------
   PXA_STATRES PXA_CMD27()
   {
    PXA_ShowDebugMessage("\r\nSend CMD27   PROGAM_CSD...\r\n");
    XllpMmcSdInts(g_pPXA_MmcReg,XLLP_MMC_I_MASK_ENDCMDRES);
    //
    XllpMmcSdSetupCmd(g_pPXA_MmcReg, XLLP_MMC_SPICMD27, 0x00, 64,0);

    //得到返回状态
       while (!(g_pPXA_MmcReg->MMC_STAT & XLLP_STAT_ENDCMDRES))
    {
        PXA_Sleep(PXAMMC_WAITTIME);
    }
    return PXA_CMD_RESP();
   }


   //------------------------------------
   ///@brief 11.1 ACMD6 .//SET_BUSWIDTH
   //------------------------------------
   PXA_STATRES PXA_ACMD6()
   {
    PXA_ShowDebugMessage("\r\nSend ACMD6...\r\n");
    XllpMmcSdInts(g_pPXA_MmcReg,XLLP_MMC_I_MASK_ENDCMDRES);
    XllpMmcSdSetupCmd(g_pPXA_MmcReg, XLLP_SD_ACMD6, 0x00000002, 64,0);
    //得到返回状态
       while (!(g_pPXA_MmcReg->MMC_STAT & XLLP_STAT_ENDCMDRES))
    {
        PXA_Sleep(PXAMMC_WAITTIME);
    }
    return PXA_CMD_RESP();
```

```
}

//-----------------------------------
///@brief 1.7 CMD16 .//SET_BLOCKLEN
//-----------------------------------
PXA_STATRES PXA_CMD16()
{
 PXA_ShowDebugMessage("\r\nSend CMD16...\r\n");
 XllpMmcSdInts(g_pPXA_MmcReg,XLLP_MMC_I_MASK_ENDCMDRES);
 //参数定义，参数为块的大小=512字节，处理的是MMC卡
 XllpMmcSdSetupCmd(g_pPXA_MmcReg, XLLP_MMC_SPICMD16, PXA_BLOCK_SIZE, 64,0);
 //得到返回状态
     while (!(g_pPXA_MmcReg->MMC_STAT & XLLP_STAT_ENDCMDRES))
 {
     PXA_Sleep(PXAMMC_WAITTIME);
 }
 return PXA_CMD_RESP();
}

//-----------------------------------
///@brief 1.7 CMD17 .//READ_SINGLE_BLOCK
///@param dwSectorIndex = 扇区序号
///@param pBlockBuffer = 512字节的块缓冲区
//-----------------------------------
PXA_STATRES PXA_CMD17(DWORD dwSectorIndex,BYTE* pBlockBuffer)
{
 WORD    nReadDataCount = 0x00;//读的数据个数
 DWORD dwAddress = dwSectorIndex*PXA_BLOCK_SIZE;//得到数据开始地址

 //PXA_ShowDebugMessage("\r\nSend CMD17...\r\n");
 XllpMmcSdInts(g_pPXA_MmcReg,XLLP_MMC_I_MASK_ENDCMDRES);

 //只传送一块数据
     XllpMmcSdSetupXCmd(g_pPXA_MmcReg, PXA_BLOCK_SIZE, PXA_BLOCK_SIZE ,
0xffff);
 //查看第 dwSectorIndex 扇区
 XllpMmcSdSetupCmd(g_pPXA_MmcReg, XLLP_MMC_SPICMD17, dwAddress, 64,0);

 //设置读请求寄存器
 XllpMmcSdInts(g_pPXA_MmcReg, XLLP_MMC_I_MASK_RXFIFORDREQ);

 //等待数据传送完毕
 do
 {
     if((g_pPXA_MmcReg->MMC_I_REG & XLLP_MMC_I_REG_RXFIFORDREQ) ==
XLLP_MMC_I_REG_RXFIFORDREQ)
     {

         int i;
```

```
        //pBlockBuffer
        for( i = 0x00 ; i < 8 ; i++)
        {
            DWORD  dwRXData = g_pPXA_MmcReg->MB_RXFIFO;
            pBlockBuffer[nReadDataCount] = (BYTE)(dwRXData&0x000000FF);
            nReadDataCount++;
            pBlockBuffer[nReadDataCount] =
(BYTE)( (dwRXData&0x0000FF00)>>8 );
            nReadDataCount++;
            pBlockBuffer[nReadDataCount] =
(BYTE)( (dwRXData&0x00FF0000)>>16 );
            nReadDataCount++;
            pBlockBuffer[nReadDataCount] =
(BYTE)( (dwRXData&0xFF000000)>>24 );
            nReadDataCount++;
        }
    }

    }while ( (!(g_pPXA_MmcReg->MMC_STAT & XLLP_STAT_DATATRANDONE))||
(nReadDataCount< PXA_BLOCK_SIZE) );


  //得到返回状态
    while (!(g_pPXA_MmcReg->MMC_STAT & XLLP_STAT_ENDCMDRES))
  {
  //  PXA_Sleep(50);
  }
  //PXA_ShowDebugMessage("\r\n");
  //PXA_ShowDebugMessage("ReadDataCount=%d\r\n",nReadDataCount);
  //ShowBuffer(pBlockBuffer);


  {
    static  UINT16      cmd17res[8];
    static  PXA_STATRES     cmd17statusRes ;
    /////////////////////////////////////////////
    XllpMmcXtractResp(g_pPXA_MmcReg, cmd17res, XLLP_MMC_MMCSPIR1);
    return  cmd17statusRes;
  }

  return PXA_CMD_RESP();
 }



 ////////////////////////////////////////////////////
```

```
//FAT文件操作

///@brief 2.1 判断FAT类型,并返回根目录扇区序号和簇包含的扇区个数
/// (dwFirstRootDirSecIndex-nFATSecIndex )/2 就是FAT所占用的扇区个数,
///@param [in] pBPB
///@param [in/out] dwFirstRootDirSecIndex = 返回根目录扇区序号
///@param [in/out] nSectsPerClust = 返回每个文件簇包含的扇区个数
///@param [in/out] nFATSecIndex = 返回FAT扇区序号
///@return 返回FAT文件系统类型

PXA_FAT_TYPE CheckFATType(BIOSPB* pBPB, DWORD *dwFirstRootDirSecIndex, WORD*
nSectsPerClust, WORD* nFATSecIndex)
{
 WORD    FATSz = 0x00;
 WORD    TotSec = 0x00;
 DWORD   DataSec = 0x00;
 DWORD   RootDirSectors = 0x00;
 DWORD   FirstDataSector = 0x00;
 DWORD   CountofClusters = 0x00;
 DWORD   FirstRootDirSecNum = 0x00;

 ///////////////////////////////////////
 RootDirSectors = ( (pBPB->NumRootEntries*32) +
     (pBPB->BytesPerSect-1) )/pBPB->BytesPerSect;

 if( pBPB->SectsPerFAT != 0)
 {
     FATSz = pBPB->SectsPerFAT;//不是FAT32
 }
 if( pBPB->SectsPerPart != 0x00)
 {
     TotSec = pBPB->SectsPerPart;//BPB_TotSec16
 }
 //FAT16文件系统根目录所在的扇区位置 FirstRootDirSecNum ，根目录占用32扇区
 FirstRootDirSecNum = pBPB->RsvdSects + pBPB->NumFATs*FATSz;

 FirstDataSector = pBPB->RsvdSects + pBPB->NumFATs*FATSz + RootDirSectors;//
开始扇区
 DataSec = TotSec -
     (pBPB->RsvdSects + pBPB->NumFATs*FATSz + RootDirSectors);//总的数据扇区数

 CountofClusters = DataSec/pBPB->SectsPerClust;//总文件簇
 ///////////////////////////////////////////////////////
 *nFATSecIndex = pBPB->RsvdSects;//FAT扇区序号
 *dwFirstRootDirSecIndex = FirstRootDirSecNum;
 *nSectsPerClust = pBPB->SectsPerClust;
 g_DataSectorStart = FirstDataSector;//开始数据扇区
 g_SectorsPerClust = pBPB->SectsPerClust;//簇包含扇区个数

 if( CountofClusters < 4085 )
 {
     // Volume is FAT12
```

```
        return PXA_FAT_TYPE_12;
    } else if( CountofClusters < 65525)
    {
        // Volume is FAT16
        return PXA_FAT_TYPE_16;

    } else
    {
        if(pBPB->SectsPerFAT != 0)
        {
            return PXA_FAT_TYPE_16;
        }
        //Volue is FAT32
        return PXA_FAT_TYPE_32;
    }
    return PXA_FAT_TYPE_NULL;

    /*
    CString str;
    str.Format("簇包含扇区数=%d FirstRootDirSecNum =%d FirstDataSector =%d
RootDirSectors=%d Clus = %d",

    pBPB->SectsPerClust,FirstRootDirSecNum,FirstDataSector,RootDirSectors,Coun
tofClusters);
    AfxMessageBox(str);

    GetRootDirFile(FirstRootDirSecNum,32,pBPB->SectsPerClust);

    */
    }

    ///@brief 2.2 初始化FAT数据
    ///@param pBlock=FAT数据
    ///@param PFT = FAT文件系统类型
    void InitFATData(BYTE* pBlock,PXA_FAT_TYPE PFT)
    {
    g_PXA_FAT_TYPE = PFT;
    memcpy(g_PXA_FAT,pBlock,512);
    }

    ///@brief 2.3 得到下一个文件簇
    ///@param nCurrentClust = 当前文件簇( >=2 )
    ///@return 返回下一个文件簇，如果是 0xFFFFFFFF 则没有下一个了
    DWORD   GetNextClust(DWORD nCurrentClust)
    {
    //现在只支持 FAT12/FAT16
    if( nCurrentClust<=1)   return nCurrentClust;
    if( PXA_FAT_TYPE_12==g_PXA_FAT_TYPE)
    {
        WORD    *pFAT = NULL;
        WORD    FATClust = 0x00;//当前簇指向的下一个文件簇序号
```

```
        WORD    nOffset = (WORD)nCurrentClust&0x0001;
        DWORD   FATOffset =  nCurrentClust +nOffset;
        if( nCurrentClust>=0x0FFF)  return 0xFFFFFFFF;
        pFAT = (WORD*)&g_PXA_FAT[FATOffset];//最大512
        FATClust = pFAT[0];//
        FATClust = nOffset ? (FATClust&0x0FFF) : ( (FATClust>>4)&0x0FFFF ) ;
        return FATClust;
    }
    else if(PXA_FAT_TYPE_16==g_PXA_FAT_TYPE)
    {
        WORD    *pFAT = NULL;
        WORD    FATClust = 0x00;//当前簇指向的下一个文件簇序号
        if( nCurrentClust >= 0xFFFF)    return 0xFFFFFFFF;
        pFAT = (WORD*)&g_PXA_FAT[nCurrentClust*2];//最大512
        FATClust = pFAT[0];//
        return FATClust;
    }
    else
    {
        return 0xFFFFFFFF;
    }

}

///@brief 2.4 文件簇序号转换成扇区序号
///@param dwClusterIndex=文件簇序号
///@param ClusSecCount = 文件簇包含的扇区个数
///@return 扇区序号
DWORD   FileClusterToSector(DWORD dwClusterIndex, WORD ClusSecCount)
{
 //g_DataSectorStart = 0x00;//文件数据扇区开始位置。由得到FAT分区类型时得到
 DWORD ClusterOffset = dwClusterIndex - 2;//因为最开始的就是第2簇，所以要减掉2
 //具体扇区位置
 DWORD SecOffset = g_DataSectorStart + dwClusterIndex* ClusSecCount;//其中32
表示每簇包含的扇区数
 return SecOffset;
}


//FILEDIRECTORY;//文件目录结构
//得到根目录扇区的第一个扇区中的我们需要的文件(NK.BIN\NK.INI)目录结构
//static UINT32           g_NKFindFlags = 0x00;//是否找到对应的文件标志
//static FILEDIRECTORY    g_NK_INI_DIR;//NK  配置文件 NK.INI 目录信息
//static FILEDIRECTORY    g_NK_BIN_DIR;//NK  镜像文件 NK.BIN 目录信息

///@brief 2.5 计算根目录和对应的文件 FILEDIRECTORY
///@param    pDirSecBuffer = 根目录扇区缓冲区
void GetRootDirFile(BYTE* pDirSecBuffer)
{
 int nFileCount = 16;//每扇区可以用16个文件
 int j = 0x00;
```

```
    FILEDIRECTORY* pFile = (FILEDIRECTORY*)pDirSecBuffer;
    for( j = 0x00 ; j < nFileCount; j++)
    {
        if( pFile[j].FileAttr==0x0F ) continue;//长文件名，不支持
        if( pFile[j].FileAttr==0x10 ) continue;//目录，不支持
        if( pFile[j].FileAttr==0x08 ) continue;//卷标，不支持

        //已经删除的，不支持

    if((pFile[j].FileName[0]==0xE5)||(pFile[j].FileName[0]==0x00)||(pFile[j].F
ileName[0]==0x05))
        {
            continue;
        }
        ////////////////////////////////////
        if((pFile[j].FileName[0]==pFile[j].FileName[1])
            &&(pFile[j].FileName[0]==pFile[j].FileName[2])
            &&(pFile[j].FileName[0]==0x00))
        {
            goto SEARCH_FINISHED;//已经没有其他文件目录
        }
        ////////////////////////////////////////////
        //g_RootFile[g_RootFileCount];//找 NK.INI 文件

    if( (pFile[j].FileName[0]=='N')&&(pFile[j].FileName[1]=='K')&&(pFile[j].Fi
leName[2]==0x20) )
        {

    if( (pFile[j].ExtName[0]=='I')&&(pFile[j].ExtName[1]=='N')&&(pFile[j].ExtN
ame[2]=='I') )
            {
                memcpy(&g_NK_INI_DIR,&pFile[j],32);//找到了INI文件
                //FIND_NKBIN_FLAGS
                g_NKFindFlags |= FIND_NKINI_FLAGS;//
            }
        }

        //g_RootFile[g_RootFileCount];//找 NK.BIN 文件

    if( (pFile[j].FileName[0]=='N')&&(pFile[j].FileName[1]=='K')&&(pFile[j].Fi
leName[2]==0x20) )
        {

    if( (pFile[j].ExtName[0]=='B')&&(pFile[j].ExtName[1]=='I')&&(pFile[j].ExtN
ame[2]=='N') )
            {
                memcpy(&g_NK_BIN_DIR,&pFile[j],32);//找到了BIN文件
                //FIND_NKBIN_FLAGS
                g_NKFindFlags |= FIND_NKBIN_FLAGS;//
            }
        }
        //可以在此添加更多的文件信息查找
```

```
}//end for j
//////////////////////////////////////////////
SEARCH_FINISHED:
;
}

///@brief 2.6得到文件目录 FIND_NKBIN_DIR
///@param nFileIndex=指定的文件类型序号
///@return 返回具体的文件目录结构指针 FILEDIRECTORY
FILEDIRECTORY*  GetFileDir(DWORD nFileIndex)
{
 FILEDIRECTORY* pDir = NULL;
 if( (g_NKFindFlags&nFileIndex) != nFileIndex ) return NULL;
 ////////////////////////////////////////////////////////////
 switch(nFileIndex)
 {
 case FIND_NKBIN_FLAGS:
     pDir = &g_NK_BIN_DIR;
     break;
 case FIND_NKINI_FLAGS:
     pDir = &g_NK_INI_DIR;
     break;
 default:
     break;
 }
 return pDir;
}

///--------------------------------------------------------------------
///@brief 3. 读指定扇区的数据
///@param dwSectorIndex = 扇区开始序号
///@param nSecCount = 扇区个数
///@param pBlockBuffer = 块缓冲区地址
///--------------------------------------------------------------------
BOOL PXA_ReadSectors(DWORD dwSectorIndex,WORD nSecCount ,BYTE* pBlockBuffer)
{
 WORD i;
 PXA_STATRES status;
 for( i = 0x00 ; i < nSecCount ; i++)
 {
     status = PXA_CMD17(dwSectorIndex+i,&pBlockBuffer[i*512]);//读
 //  PXA_ShowDebugMessage("Info :  CMD17 status = 0x%x\r\n",status.status);

 }
 return TRUE;

}
```

LCD显示驱动源码如下所示：

```
//--------------------------------------------------------------------------
//
//          这是一个LCD显示驱动的代码
//   2007-04-06 中国重庆
//
//--------------------------------------------------------------------------
#include <windows.h>
#include <nkintr.h>
#include <bulverde.h>
#include <mainstoneii.h>
#include <oal_memory.h>
#include <pcireg.h>
#include <fmd.h>
#include <xllp_pccardsocket.h>
#include <xllp_lcd.h>
#include <xllp_cpdvm.h>


#include <bsp.h>
#include "loader.h"

////////////////////////////////////////////
extern const WORD CEStartBitmap[];

extern int OEMReadDebugByte();
extern void TestSDMMC();

////////////////////////////////////////////

///@brief 测试LCD
void PXA_TestLCD();
void EbootInitLcd();
void LightupBackLight();
void DispStartBitmap(volatile BYTE *pDisplay, BYTE *pData);

///@brief 初始化系统通过Eboot启动NK
void PXA_InitEBootForNK();

//////////////////////////////////
static   DWORD   g_dwBitmapAddr = 0x00;
///////////////////////////////////////
//系统定义相关的Image开始位置
#define PXA_RAM_IMAGE_START          0x1AA00000 //0x80030000 +
(offset)0x3A9D0000 =0xBAA00000
#define BOOT_FLASH_BASE_U_VIRTUAL    0xA0000000
#define SDRAM_BASE_PHYSICAL           0xA0000000

#define FLASH_CFG_START
 (BOOT_FLASH_BASE_U_VIRTUAL+0x00040000)//size = 0x00040000
#define LCD_FLAG                 0xEFCAF4CD //LCD
```

```
   #define DMA_DESC_SIZE                         (0x20)
   #define DISPLAY_DMA_OFFSET                    (0x0)

   //#define MEM_BASE_PHYSICAL                      (SDRAM_BASE_PHYSICAL +
0x03A00000)//0xA3A00000
   #define MEM_BASE_PHYSICAL                     (SDRAM_BASE_PHYSICAL +
0x02714000)//0xA3A00000


   #define IM_STORAGE_BASE_PHYSICAL              0x5C000000
//BULVERDE_BASE_REG_PA_IMSTORAGE

   //#define LCD_MEM_BASE_PHYSICAL                  IM_STORAGE_BASE_PHYSICAL
   #define LCD_MEM_BASE_PHYSICAL                 MEM_BASE_PHYSICAL

   #define DMA_CHANNEL_0_FRAME_DESCRIPTOR_BASE_PHYSICAL
(LCD_MEM_BASE_PHYSICAL+DISPLAY_DMA_OFFSET)  // 00-1F
   #//define DMA_CHANNEL_0_FRAME_DESCRIPTOR_BASE_VIRTUAL
(LCD_MEM_BASE_U_VIRTUAL+DISPLAY_DMA_OFFSET)

   #define DMA_CHANNEL_1_FRAME_DESCRIPTOR_BASE_PHYSICAL
(DMA_CHANNEL_0_FRAME_DESCRIPTOR_BASE_PHYSICAL+DMA_DESC_SIZE)    // 20-3F
   //#define DMA_CHANNEL_1_FRAME_DESCRIPTOR_BASE_VIRTUAL
(DMA_CHANNEL_0_FRAME_DESCRIPTOR_BASE_VIRTUAL+DMA_DESC_SIZE)

   #define DMA_CHANNEL_0_ALT_FRAME_DESCRIPTOR_BASE_PHYSICAL
(DMA_CHANNEL_1_FRAME_DESCRIPTOR_BASE_PHYSICAL+DMA_DESC_SIZE)    // 40-5F
   //#define DMA_CHANNEL_0_ALT_FRAME_DESCRIPTOR_BASE_VIRTUAL
(DMA_CHANNEL_1_FRAME_DESCRIPTOR_BASE_VIRTUAL+DMA_DESC_SIZE)

   #define PALETTE_FRAME_DESCRIPTOR_BASE_PHYSICAL
(DMA_CHANNEL_0_ALT_FRAME_DESCRIPTOR_BASE_PHYSICAL+DMA_DESC_SIZE)    // 60-7F
   //#define PALETTE_FRAME_DESCRIPTOR_BASE_VIRTUAL
(DMA_CHANNEL_0_ALT_FRAME_DESCRIPTOR_BASE_VIRTUAL+DMA_DESC_SIZE)

   #define SDRAM_DMA_CHANNEL_2_Y_FRAME_DESCRIPTOR_BASE_PHYSICAL
    (PALETTE_FRAME_DESCRIPTOR_BASE_PHYSICAL+DMA_DESC_SIZE)     // 80 - 9F
   //#define SDRAM_DMA_CHANNEL_2_Y_FRAME_DESCRIPTOR_BASE_VIRTUAL
    (PALETTE_FRAME_DESCRIPTOR_BASE_VIRTUAL+DMA_DESC_SIZE)

   #define SDRAM_DMA_CHANNEL_3_Cb_FRAME_DESCRIPTOR_BASE_PHYSICAL
    (SDRAM_DMA_CHANNEL_2_Y_FRAME_DESCRIPTOR_BASE_PHYSICAL+DMA_DESC_SIZE) // A0
- BF
   //#define SDRAM_DMA_CHANNEL_3_Cb_FRAME_DESCRIPTOR_BASE_VIRTUAL
    (SDRAM_DMA_CHANNEL_2_Y_FRAME_DESCRIPTOR_BASE_VIRTUAL+DMA_DESC_SIZE)

   #define SDRAM_DMA_CHANNEL_4_Cr_FRAME_DESCRIPTOR_BASE_PHYSICAL
    (SDRAM_DMA_CHANNEL_3_Cb_FRAME_DESCRIPTOR_BASE_PHYSICAL+DMA_DESC_SIZE) // C0
- DF
   //#define SDRAM_DMA_CHANNEL_4_Cr_FRAME_DESCRIPTOR_BASE_VIRTUAL
```

```
    (SDRAM_DMA_CHANNEL_3_Cb_FRAME_DESCRIPTOR_BASE_VIRTUAL+DMA_DESC_SIZE)

   #define DMA_PALETTE_SIZE             (0x400)
   #define PALETTE_BUFFER_BASE_PHYSICAL
(SDRAM_DMA_CHANNEL_4_Cr_FRAME_DESCRIPTOR_BASE_PHYSICAL+DMA_DESC_SIZE)  //
E0-4DF
   //#define PALETTE_BUFFER_BASE_VIRTUAL
(SDRAM_DMA_CHANNEL_4_Cr_FRAME_DESCRIPTOR_BASE_VIRTUAL+DMA_DESC_SIZE)

   #define FRAME_BUFFER_0_BASE_PHYSICAL
(PALETTE_BUFFER_BASE_PHYSICAL+DMA_PALETTE_SIZE)      // 4E0-BBCDF
   //#define FRAME_BUFFER_0_BASE_VIRTUAL
(PALETTE_BUFFER_BASE_VIRTUAL+DMA_PALETTE_SIZE)

   ///////////////////////////////////////
   #define NEW_FREQ104M (13000*8)  //104MHz
   void PXA_EXTest()
   {
    DWORD LMult,NMult;
    unsigned int CLKCFG = 0x00;
        volatile XLLP_CLKMGR_T *p_CLKRegs;
    XllpXSC1FreqChange(NEW_FREQ104M);

    CLKCFG = XllpXSC1ReadCLKCFG( );
        p_CLKRegs = (XLLP_CLKMGR_T *) OALPAtoVA(BULVERDE_BASE_REG_PA_CLKMGR,
FALSE);//FALSE=U TUE=V

        LMult = p_CLKRegs->ccsr & 0x0000001F;
        NMult = (p_CLKRegs->ccsr & 0x00000380) >> 7;

        KITLOutputDebugString("****** CLKRegs->cccr=0x%x CLKCFG=0x%x L=0x%x
N=0x%x...\r\n",p_CLKRegs->cccr,CLKCFG, LMult ,NMult);

   }

   void PXA_TestLCD()
   {
    BYTE*   pDisplayBuffer = NULL;
    BYTE*   pSourceBuffer = NULL;

    //g_dwBitmapAddr        = (DWORD)StartBitmap320x240  - (RAM_IMAGE_START -
0x40000) + BOOT_FLASH_BASE_U_VIRTUAL;
    //g_dwBitmapAddr        = (DWORD)StartBitmap320x240 ;// -
PXA_RAM_IMAGE_START;

    //  pDisplayBuffer = (BYTE *)FRAME_BUFFER_0_BASE_VIRTUAL;
    pDisplayBuffer = (BYTE
*)OALPAtoVA(FRAME_BUFFER_0_BASE_PHYSICAL,FALSE);////FALSE=U TUE=V

     //   pSourceBuffer  = (BYTE *)g_dwBitmapAddr;
    PXA_EXTest();
```

```
    //  KITLOutputDebugString("****** StartBitmap320x240=0x%x
pDisplayBuffer=0x%x...\r\n", (DWORD)StartBitmap320x240, (DWORD)pDisplayBuffer);

    ///@brief 初始化系统通过Eboot启动NK
    PXA_InitEBootForNK();
    return ;

        #if 0
        {
        WORD row,col, filled;
        KITLOutputDebugString(" selecte #if 0 ...\r\n");
        for (row = 0; row < SCREENHEIGHT; row++)
        {
            filled = row;
            for (col = 0; col < SCREENWIDTH; col++)
            {
                *(WORD *)(pDisplayBuffer + (row * SCREENWIDTH + col) * 2) =
filled++;
            }
        }
        }
        #else
        KITLOutputDebugString(" selecte #if 1 ...\r\n");

   //  if (   ((DWORD)pSourceBuffer > BOOT_FLASH_BASE_U_VIRTUAL)
   //          && ((DWORD)pSourceBuffer < FLASH_CFG_START) )
        {
            KITLOutputDebugString(" selecte #if 1 ...in range \r\n");
            DispStartBitmap(pDisplayBuffer, pSourceBuffer);

            *(volatile DWORD *)pDisplayBuffer = LCD_FLAG;
        }

        #endif

        EbootInitLcd();

    /* Light up backlight. */
    LightupBackLight();

    }

   #define SCREENWIDTH      640
   #define SCREENHEIGHT     480
   #define DISPLAYTYPE      PD064VT2

   void EbootInitLcd()
   {
       XLLP_LCD_T XllpLCD;
       volatile LCD_FRAME_DESCRIPTOR   *frameDescriptorCh0fd1=NULL;
```

```
    volatile LCD_FRAME_DESCRIPTOR *frameDescriptorCh0fd2=NULL;
    volatile LCD_FRAME_DESCRIPTOR *frameDescriptorCh1=NULL;
    volatile LCD_FRAME_DESCRIPTOR *frameDescriptorPalette=NULL;
    volatile LCD_FRAME_DESCRIPTOR *frameDescriptorTemp=NULL;
    volatile LCD_PALETTE          *v_pPaletteBuffer=NULL;

 /*
    frameDescriptorCh0fd1 = (volatile LCD_FRAME_DESCRIPTOR
*)(DMA_CHANNEL_0_FRAME_DESCRIPTOR_BASE_VIRTUAL);
    frameDescriptorCh0fd2 = (volatile LCD_FRAME_DESCRIPTOR
*)(DMA_CHANNEL_0_ALT_FRAME_DESCRIPTOR_BASE_VIRTUAL);
    frameDescriptorCh1 = (volatile LCD_FRAME_DESCRIPTOR
*)(DMA_CHANNEL_1_FRAME_DESCRIPTOR_BASE_VIRTUAL);
    frameDescriptorPalette = (volatile LCD_FRAME_DESCRIPTOR
*)(PALETTE_FRAME_DESCRIPTOR_BASE_VIRTUAL);
    v_pPaletteBuffer=(volatile LCD_PALETTE *)(PALETTE_BUFFER_BASE_VIRTUAL);
 */

    frameDescriptorCh0fd1 = (volatile LCD_FRAME_DESCRIPTOR
*)OALPAtoVA(DMA_CHANNEL_0_FRAME_DESCRIPTOR_BASE_PHYSICAL,FALSE);
    frameDescriptorCh0fd2 = (volatile LCD_FRAME_DESCRIPTOR
*)OALPAtoVA(DMA_CHANNEL_0_ALT_FRAME_DESCRIPTOR_BASE_PHYSICAL,FALSE);
    frameDescriptorCh1 = (volatile LCD_FRAME_DESCRIPTOR
*)OALPAtoVA(DMA_CHANNEL_1_FRAME_DESCRIPTOR_BASE_PHYSICAL,FALSE);
    frameDescriptorPalette = (volatile LCD_FRAME_DESCRIPTOR
*)OALPAtoVA(PALETTE_FRAME_DESCRIPTOR_BASE_PHYSICAL,FALSE);
    v_pPaletteBuffer=(volatile LCD_PALETTE
*)OALPAtoVA(PALETTE_BUFFER_BASE_PHYSICAL,FALSE);

 //BULVERDE_BASE_REG_PA_GPIO
  /*
    XllpLCD.GPIO = (XLLP_VUINT32_T *) GPIO_BASE_U_VIRTUAL;
    XllpLCD.CLKMan = (XLLP_VUINT32_T *) (PERIF_BASE_U_VIRTUAL + CLK_OFFSET);
    XllpLCD.LCDC = (XLLP_VUINT32_T *) LCD_BASE_U_VIRTUAL;
  */

    XllpLCD.GPIO = (XLLP_VUINT32_T *) OALPAtoVA(BULVERDE_BASE_REG_PA_GPIO,
FALSE);//FALSE=U TUE=V
    XllpLCD.CLKMan = (XLLP_VUINT32_T *)
OALPAtoVA(BULVERDE_BASE_REG_PA_CLKMGR, FALSE);//FALSE=U TUE=V
    XllpLCD.LCDC = (XLLP_VUINT32_T *) OALPAtoVA(BULVERDE_BASE_REG_PA_LCD,
FALSE);//FALSE=U TUE=V;//


    XllpLCD.BPP = BPP_16;
    XllpLCD.DisplayType = DISPLAYTYPE;
    XllpLCD.FrameBufferWidth  = SCREENWIDTH;
    XllpLCD.FrameBufferHeight = SCREENHEIGHT;

    XllpLCD.PixelDataFormat = PDFOR_00;
```

```
        XllpLCD._FRAME_BUFFER_BASE_PHYSICAL = FRAME_BUFFER_0_BASE_PHYSICAL;
        XllpLCD._PALETTE_BUFFER_BASE_PHYSICAL = PALETTE_BUFFER_BASE_PHYSICAL;
        XllpLCD._DMA_CHANNEL_0_FRAME_DESCRIPTOR_BASE_PHYSICAL =
DMA_CHANNEL_0_FRAME_DESCRIPTOR_BASE_PHYSICAL;
        XllpLCD._DMA_CHANNEL_1_FRAME_DESCRIPTOR_BASE_PHYSICAL =
DMA_CHANNEL_1_FRAME_DESCRIPTOR_BASE_PHYSICAL;
        XllpLCD._DMA_CHANNEL_0_ALT_FRAME_DESCRIPTOR_BASE_PHYSICAL =
DMA_CHANNEL_0_ALT_FRAME_DESCRIPTOR_BASE_PHYSICAL;
        XllpLCD._PALETTE_FRAME_DESCRIPTOR_BASE_PHYSICAL =
PALETTE_FRAME_DESCRIPTOR_BASE_PHYSICAL;
        XllpLCD.frameDescriptorCh0fd1 = frameDescriptorCh0fd1;
        XllpLCD.frameDescriptorCh0fd2 = frameDescriptorCh0fd2;
        XllpLCD.frameDescriptorCh1 = frameDescriptorCh1;
        XllpLCD.frameDescriptorPalette = frameDescriptorPalette;
        XllpLCD.frameDescriptorTemp = frameDescriptorTemp;
        XllpLCD.CurrentPage = 0;

        // Initialize the LCD Controller
        XllpLCDInit(&XllpLCD);
    }

    typedef struct
    {
        volatile unsigned long   pwmcr0;
        volatile unsigned long   pwmdcr0;
        volatile unsigned long   pwmpcr0;
        volatile unsigned long   reserved;
        volatile unsigned long   pwmcr2;
        volatile unsigned long   pwmdcr2;
        volatile unsigned long   pwmpcr2;
    }PWM_REGS, *PPWM_REGS;

    void LightupBackLight()
    {

    /*
    PPWM_REGS pTPwm = (PPWM_REGS)OALPAtoVA(BULVERDE_BASE_REG_PA_PWM0_2,FALSE);

    XLLP_GPIO_T *pGpio = (XLLP_GPIO_T *)OALPAtoVA(BULVERDE_BASE_REG_PA_GPIO,
FALSE);//FALSE=U TUE=V
    //
    P_XLLP_OST_T pOSTRegs = (P_XLLP_OST_T)OALPAtoVA(BULVERDE_BASE_REG_PA_OST,
FALSE);//FALSE=U TUE=V

    P_IND_CTRL pIndCtrl = (P_IND_CTRL)INDICATOR_LIGHT_BASE_U_VIRTUAL;

    int ival= pTPwm->pwmdcr0 & 0x3FF ;
    while(ival-- > 0x0)
    {
        pTPwm->pwmcr0 = 0xf;
```

```
        pTPwm->pwmdcr0 = ival;
        pTPwm->pwmpcr0 = 0x1f;
        XllpOstDelayMilliSeconds(pOSTRegs,20);
    }

    */
    }


    struct COMPRESS_HEADER
    {
        DWORD   type    :4;
        DWORD   width   :10;
        DWORD   height  :10;
        DWORD   colors  :8;
        DWORD   size;
    };


    #define N         4096   /* size of ring buffer */
    #define F           18   /* upper limit for match_length */
    #define THRESHOLD    2   /* encode string into position and length
                             if match_length is greater than this */

    BYTE text_buf[N + F - 1];   /* ring buffer of size N, with extra F-1 bytes to
facilitate string comparison */

    void LZSS_Decode(BYTE* In, volatile BYTE* Out, DWORD InSize, DWORD OutSize)
    {
    DWORD i, j, k, r;
    DWORD flags;
    DWORD dwInBufPos = 0;
        DWORD dwOutBufPos = 0;

        //EdbgOutputDebugString("In @ 0x%x, Out @ 0x%x, In size = %d, Out size =
%d \r\n",
        //                              In, Out, InSize, OutSize);

        if ( NULL == In || NULL == Out || 0 == InSize || 0 == OutSize )
            return;
    for (i = 0; i < N - F; i++)
        text_buf[i] = ' ';
    r = N - F;
    flags = 0;
    for ( ; ; )
    {
        if (((flags >>= 1) & 256) == 0)
        {
            if (dwInBufPos >= InSize)
                break;
            flags = In[dwInBufPos++] | 0xff00;     /* uses higher byte cleverly
```

```
*/
        }                               /* to count eight */
        if (flags & 1)
        {
            if (dwInBufPos >= InSize)
                break;
            Out[dwOutBufPos++] = In[dwInBufPos];
            text_buf[r++] = In[dwInBufPos++];
            r &= (N - 1);
        }
        else
        {
            if (dwInBufPos >= InSize - 1)
                break;
            i = In[dwInBufPos++];
            j = In[dwInBufPos++];
            i |= ((j & 0xf0) << 4);
            j = (j & 0x0f) + THRESHOLD;
            for (k = 0; k <= j; k++)
            {
                BYTE c = text_buf[(i + k) & (N - 1)];
                Out[dwOutBufPos++] = c;
                text_buf[r++] = c;
                r &= (N - 1);
            }
        }
    }
    //EdbgOutputDebugString("In Position = %d, Out Position = %d\r\n",
dwInBufPos, dwOutBufPos);
  }


void DispStartBitmap(volatile BYTE *pDisplay, BYTE *pData)
{
    volatile struct COMPRESS_HEADER *pHeader;
    BYTE *pBitmap;
    DWORD InSize, OutSize;

    if ( (NULL == pData) || (NULL == pDisplay) )
        return;

    pHeader = (volatile struct COMPRESS_HEADER *)pData;
    pBitmap = (BYTE *)((DWORD)pData + sizeof(struct COMPRESS_HEADER));

    InSize  = pHeader->size;
    OutSize = pHeader->width * pHeader->height * pHeader->colors / 8;

    if ( (0 == InSize) || (0 == OutSize) )
        return;
```

```
    switch (pHeader->type)
    {
        case 0:              //no compress
            memcpy((PVOID)pDisplay, (PVOID)pBitmap, OutSize);
            break;

        case 1:              //lzss
            LZSS_Decode(pBitmap, pDisplay, InSize, OutSize);
            break;

        default:
            break;
    }
}

//---------------------------------------------------------------
///@brief 初始化系统通过Eboot启动NK
//extern const WORD CEStartBitmap[];
typedef struct CEBITMAPHEAD
{
 WORD    StartX;          //开始位置X
 WORD    StartY;          //开始位置Y
 WORD    Width;           //图片宽度
 WORD    Height;          //图片高度
 WORD    Reserved[4];  //保留
}CEBITMAPHEAD;//CEStartBitmap位图头信息

static   WORD    g_nCurrentGuage = 0x00;//开始位置
static   BYTE*   g_pDisplayBuffer=NULL;  //显存地址
//进度条颜色定义
static   WORD    g_nRGB565Guage = 0xFFE0;//黄色
static   WORD    g_nRGB565Init = 0x0014; //蓝色

//----------------------------------------------------
//PXA_InitGuage() 初始化进度条信息 RGB=0x0014
//----------------------------------------------------
void PXA_InitGuage()
{
 //以下显示进度条信息 guage 进度指示
 //进度条长度为宽度的10/16 .按照每个显示2.5% 即采用40个显示进度表示。高度采用整体高度的
1/80
 //开始位置为整体高度的7/8位置
 int i,j;
 DWORD nOffset = (SCREENWIDTH*SCREENHEIGHT*7/8 + SCREENWIDTH*3/16)*2 ; //开
始位置
 WORD    nHeight = SCREENHEIGHT*1/80;
 WORD    nWidthSize = (SCREENWIDTH*10/16);//长度

 for( i = 0x00 ; i< nWidthSize; i++)
 {
```

```
        for( j = 0x00;  j < nHeight;j++)
        {
            g_pDisplayBuffer[SCREENWIDTH*j*2 + i*2 + nOffset+1]    =
(BYTE)(g_nRGB565Init>>8);//0x00
            g_pDisplayBuffer[SCREENWIDTH*j*2 + i*2 + nOffset] =
(BYTE)(g_nRGB565Init&0xFF); //0x14
        }
    }
  }


  //----------------------------------------------------
  ///PXA_ShowGuage 显示进度条.fPercent = 进度百分比(0--100)
  //----------------------------------------------------
  void PXA_ShowGuage(WORD nPercent)
  {
   nPercent = MIN(nPercent,100);
   if( g_nCurrentGuage >= nPercent )  return ;
   ////////////////////////////////////////////////
   //进度条长度为宽度的10/16 .高度采用整体高度的1/80 ,开始位置为整体高度的7/8位置
   {
        int i,j;
        DWORD nOffset = (SCREENWIDTH*SCREENHEIGHT*7/8 + SCREENWIDTH*3/16)*2 ; //
开始位置
        WORD    nHeight = SCREENHEIGHT*1/80;
        WORD    nWidthSize = (SCREENWIDTH*10/16);//长度
        WORD    nStart = g_nCurrentGuage*nWidthSize/100;//百分比
        WORD    nStop = nPercent*nWidthSize/100;//百分比
        g_nCurrentGuage = nPercent;//保存当前位置
        for( i = nStart ; i< nStop; i++)
        {
            for( j = 0x00;  j< nHeight;j++)
            {
                g_pDisplayBuffer[SCREENWIDTH*j*2 + i*2 + nOffset+1]    =
(BYTE)(g_nRGB565Guage>>8);//0xFF
                g_pDisplayBuffer[SCREENWIDTH*j*2 + i*2 + nOffset] =
(BYTE)(g_nRGB565Guage&0xFF); //0xE0
            }
        }
   }
  }
  //----------------------------------------------------

  //----------------------------------------------------
  //PXA_EbootReadSDNK 从SD卡读取NK.BIN信息
  //----------------------------------------------------
  void PXA_EbootReadSDNK()
  {
   //PXA_ShowGuage(offset);
   TestSDMMC();
```

```
}


//----------------------------------------
//PXA_InitEBootForNK 初始化NK显示信息
//----------------------------------------
void PXA_InitEBootForNK()
{
 DWORD offset = 0x00;
 BYTE*   pDisplayBuffer = NULL;
 WORD*   pCEBitmapSource = NULL;
 BYTE*   pBitSource = NULL;
 CEBITMAPHEAD    *pBitmapHead = NULL;
 pDisplayBuffer = (BYTE
*)OALPAtoVA(FRAME_BUFFER_0_BASE_PHYSICAL,FALSE);////FALSE=U TUE=V
 memset(pDisplayBuffer,0x00,640*480*2);
 g_pDisplayBuffer = pDisplayBuffer;//保存显存地址
 ////////////////////////////////////////////////////////////////
 //以下是显示Log标志
 pBitSource = (BYTE*)CEStartBitmap;
 pBitmapHead = (CEBITMAPHEAD*)CEStartBitmap;//得到头信息

 pCEBitmapSource = (WORD*)(pBitSource + sizeof(CEBITMAPHEAD)); //得到缓冲区地
址
 {
     int i;
     DWORD   nOffset = (SCREENWIDTH*pBitmapHead->StartY +
pBitmapHead->StartX)*2;//开始位置
     WORD    nCopySize = pBitmapHead->Width*2;//按照字节计算

     for( i = 0x00 ; i < pBitmapHead->Height;i++)
     {
        memcpy( &pDisplayBuffer[SCREENWIDTH*i*2 + nOffset ] ,
&pCEBitmapSource[i*pBitmapHead->Width],nCopySize);
     }
     //显示完成后需要重新设置进度条开始位置=0;
     g_nCurrentGuage = 0x00;//进度条开始位置
 }
 //以上是显示Log标志
 //--------------------------------------------------
 //初始化进度条信息。
 PXA_InitGuage();

     KITLOutputDebugString("...DisplayPA=0x%x  DisplayVA=0x%x...\r\n",
(DWORD)FRAME_BUFFER_0_BASE_PHYSICAL, (DWORD)pDisplayBuffer);
     EbootInitLcd();

 ////////////////////////////////////////////////////////////////
```

```
return ;//直接返回。
   KITLOutputDebugString("...Please enter A key to continue...\r\n");

while(1)
{

    while(1)
    {
        int     b = 0;
        b = OEMReadDebugByte();
        if( (b=='A') || (b=='a'))
        {
            offset++;
            break;
        }
    }
    /////////////////////////////
    PXA_ShowGuage(offset);
    if( offset >= 100) break;
    //////////////////////////////////////////
    /*
    //开始位置为整体高度的7/8位置
    {
        int i ,j;
        DWORD nOffset = (SCREENWIDTH*SCREENHEIGHT*7/8 + SCREENWIDTH*3/16)*2 ;
//开始位置
        WORD    nHeight = SCREENHEIGHT*1/80;
        WORD    nWidthSize = (SCREENWIDTH*10/16);//长度
 //     for( i = 0x00 ; i< nWidthSize; i++)
        {
            for( j = 0x00;   j < nHeight;j++)
            {//可以将offset 改成 i
                pDisplayBuffer[SCREENWIDTH*j*2 + (offset-1)*2 + nOffset+1]
= 0xFF;
                pDisplayBuffer[SCREENWIDTH*j*2 + (offset-1)*2 + nOffset] =
0xE0;
            }
            if( offset >= nWidthSize)  break;

        }
    }
    */

}

  KITLOutputDebugString("...Please enter A key to continue...\r\n");

while(1)
```

```
    {
        int      b = 0;
        b = OEMReadDebugByte();
        if( (b=='A') || (b=='a')) break;
    }

}
```

启动画面包含在 CEBmp565.c 文件当中，其中只定义了 CEStartBitmap 数组，其主要内容如下：

```
//==================================================================

//        该程序是由 BMP565.EXE 自动生成,作为PXA270的启动画面代码
//      PXA270 采用的RGB(565)方式排列数据的。
//        CEStartBitmap 数据的组织格式为前  8个WORD数据为BMP数据的
//      头信息,后面的数据为具体的BMP数据信息:
//      宽度= 280   高度= 60
//      开始宽度= 180   开始高度= 210
//      整个数组长度为= 33616 BYTEs

//                          2007-04-20 chongqing china

//==================================================================
#include <windows.h>

const WORD CEStartBitmap[] =
{
 /*开始位置X*/0X00B4, /*开始位置Y*/0X00D2, /*位图宽度W*/0X0118, /*位图高度
H*/0X003C,  /*保留1*/0X0000,/*保留2*/0X0000,/*保留3*/0X0000,/*保留4*/0X0000,
  //  高度位置 = 0 , 每行包含的像素个数 = 16
  0X0000,0X0000,0X0000,0X0000,0X0000,0X0000,0X0000,0X0000,0X0000,0X0000,
0X0000, 0X0000, 0X0000, 0X0000, 0X0000, 0X0000,
  ......//这里省略了其他像素的内容
  }
```

Eboot 程序可以通过 PB 的集成编译环境编译链接，控制文件为.bib 文件。下面是本例 Eboot 的.bib 文件。

```
;
; Copyright (c) Microsoft Corporation.  All rights reserved.
;
;
; Use of this sample source code is subject to the terms of the Microsoft
; license agreement under which you licensed this sample source code. If
; you did not accept the terms of the license agreement, you are not
; authorized to use this sample source code. For the terms of the license,
; please see the license agreement between you and Microsoft or, if applicable,
; see the LICENSE.RTF on your install media or the root of your tools
installation.
; THE SAMPLE SOURCE CODE IS PROVIDED "AS IS", WITH NO WARRANTIES.
;
```

```
;*****************************************************************************
*****
; TITLE:        EBOOT.BIB
;
; Ethernet Boot Loader Source Module
;
; Memory Map
; ----------
;
; 8000.0000 -+
;            | Bootloader Stack (64KB)
; 8001.0000 -+
;            | Bootloader RAM (64KB)
; 8002.0000 -+
;            | Bootloader Code (256KB)
; 8006.0000 -+
;            | GAP (reserved for future)
; 800F.F000 -+
;            | ARGS (4KB)
; 8010.0000 -+
;            |
;            | OS (NK) Code (48MB)
;            |
; 8310.0000 -+   **auto-size**
;            |
;            | OS (NK) RAM (79MB)
;            |
; 8800.0000 -+
;
;*****************************************************************************
*****
MEMORY

;   Name     Start     Size      Type
;   -------  --------  --------  ----
    STACK    80000000  00010000  RESERVED
    RAM      80010000  00020000  RAM
    EBOOT    80030000  00040000  RAMIMAGE
    RSVD     80070000  0008F000  RESERVED
    ARGS     800FF000  00001000  RESERVED


CONFIG
    AUTOSIZE=OFF
    COMPRESSION=OFF
    PROFILE=OFF
    KERNELFIXUPS=ON

    ROMSTART=80030000
    ROMWIDTH=32
```

```
        ROMSIZE=00040000

        ; The bootloader is fixed up to run from RAM, but is stored in flash.
ROMOFFSET
        ; adjusts the .bin file record headers to that they lie in flash (this tells
the
        ; bootloader already on the device to store the image in flash).
    IF MCPII_MODULE
        ROMOFFSET=3A9D0000
    ELSE
        ROMOFFSET=3C9D0000
    ENDIF

    MODULES

    ;   Name    Path                                                     Memory
Type
    ;   -----------------------------------------------------------------------
--------
        nk.exe  $(_TARGETPLATROOT)\target\$(_TGTCPU)\$(WINCEDEBUG)\eboot.exe
EBOOT
```

Eboot 开发完成之后，需要下载到目标开发板，对于 Eboot 文件的下载，有很多种方法：可以通过仿真器下载；也可以通过其他调试程序下载；还可以直接烧写到 Flash 中。需要说明的一点是，这些方法可能会要求不同的映像格式。在 PB 环境下，可以生成的有.sre 格式、纯二进制格式（用于直接烧写 Flash）以及和 CE 映像一样的.bin 格式。

4.3 其他硬件平台的 BootLoader

4.3.1 vivi 介绍

vivi 是当前比较流行的，专门针对 ARM9 处理器而设计的一款 Bootloader，它操作简便，同时提供了完备的命令体系。

vivi 是由韩国 Mizi 公司开发的一种 Bootloader，适合于 ARM9 处理器，支持 S3C2410x 处理器，其源代码可以在 http://www.mizi.com 网站下载。和所有的 Bootloader 一样，vivi 有两种工作模式，即启动加载模式和下载模式。当 vivi 处于下载模式时，它为用户提供一个命令行接口，通过该接口能使用 vivi 提供的一些命令集。

vivi 作为一种 Bootloader，其运行过程分成两个阶段。第一阶段的代码在 vivi/arch/s3c2410/head.s 中定义，大小不超过 10 KB，它包括从系统加电后在 0x00000000 地址开始执行的部分。这部分代码运行在 Flash 中，它包括对 S3C2410 的一些寄存器、时钟等的初始化并跳转到第二阶段执行。第二阶段的代码在 vivi\init\main.c 中，主要进行一些开发板初始化、内存映射和内存管理单元初始化等工作，最后会跳转到 boot_or_vivi()函数中，接收命令并进行处理。需要注意的是，在 Flash 中执行完内存映射后，会将 vivi 代码复制到 SDRAM 中执行。

4.3.2 U-Boot 介绍

U-Boot，全称 Universal Boot Loader，是遵循 GPL 条款的开放源码项目。从 FADSROM、8xxROM、PPCBOOT 逐步发展演化而来。其源码目录、编译形式与 Linux 内核很相似，事实上，不少 U-Boot 源码就是相应的 Linux 内核源程序的简化，尤其是一些设备的驱动程序，这从 U-Boot 源码的注释中能体现这一点。但是 U-Boot 不仅仅支持嵌入式 Linux 系统的引导，当前，它还支持 NetBSD、VxWorks、QNX、RTEMS、ARTOS、LynxOS 嵌入式操作系统。其目前要支持的目标操作系统是 OpenBSD、NetBSD、FreeBSD、4.4BSD、Linux、SVR4、Esix、Solaris、Irix、SCO、Dell、NCR、VxWorks、LynxOS、pSOS、QNX、RTEMS、ARTOS。这是 U-Boot 中 Universal 的一层含义，另外一层含义则是 U-Boot 除了支持 PowerPC 系列的处理器外，还能支持 MIPS、x86、ARM、NIOS、XScale 等诸多常用系列的处理器。这两个特点正是 U-Boot 项目的开发目标，即支持尽可能多的嵌入式处理器和嵌入式操作系统。就目前来看，U-Boot 对 PowerPC 系列处理器支持最为丰富，对 Linux 的支持最完善。其他系列的处理器和操作系统基本上是在 2002 年 11 月 PPCBOOT 改名为 U-Boot 后逐步扩充的。从 PPCBOOT 向 U-Boot 的顺利过渡，很大程度上归功于 U-Boot 的维护人德国 DENX 软件工程中心 Wolfgang Denk（以下简称 W.D）本人精湛专业水平和执着不懈的努力。

1. U-Boot 的特点

（1） 开放源码。

（2） 支持多种嵌入式操作系统内核，如 Linux、NetBSD、VxWorks、QNX、RTEMS、ARTOS、LynxOS。

（3） 支持多个处理器系列，如 PowerPC、ARM、x86、MIPS、XScale。

（4） 较高的可靠性和稳定性。

（5） 高度灵活的功能设置，适合 U-Boot 调试、操作系统不同引导要求、产品发布等。

（6） 丰富的设备驱动源码，如串口、以太网、SDRAM、FLASH、LCD、NVRAM、EEPROM、RTC、键盘等。

（7） 较为丰富的开发调试文档与强大的网络技术支持。

2. U-Boot 主要目录结构

（1） board 目标板相关文件，主要包含 SDRAM、FLASH 驱动。

（2） common 独立于处理器体系结构的通用代码，如内存大小探测与故障检测。

（3） cpu 与处理器相关的文件。如 mpc8xx 子目录下含串口、网口、LCD 驱动及中断初始化等文件。

（4） driver 通用设备驱动。如 CFI FLASH 驱动（目前对 INTEL FLASH 支持较好）。

（5） doc U-Boot 的说明文档。

（6） examples 可在 U-Boot 下运行的示例程序。如 hello_world.c、timer.c。

（7） include U-Boot 头文件；尤其 configs 子目录下与目标板相关的配置头文件是移植过程中经常要修改的文件。

（8） lib_xxx 处理器体系相关的文件，如 lib_ppc, lib_arm 目录分别包含与 PowerPC、ARM 体系结构相关的文件。

（9） net 与网络功能相关的文件目录，如 bootp、nfs、tftp。

（10） post 加电自检文件目录。尚有待于进一步完善。

（11） rtc RTC 驱动程序。

（12） tools 用于创建 U-Boot S-RECORD 和 BIN 镜像文件的工具。

3. U-Boot 支持的主要功能

（1） 系统引导。支持 NFS 挂载、RAMDISK（压缩或非压缩）形式的根文件系统。

（2） 支持 NFS 挂载、从 FLASH 中引导压缩或非压缩系统内核。

（3） 基本辅助功能。强大的操作系统接口功能；可灵活设置、传递多个关键参数给操作系统，适合系统在不同开发阶段的调试要求与产品发布。

（4） Linux 支持最为强劲。

（5） 支持目标板环境参数多种存储方式，如 FLASH、NVRAM、EEPROM。

（6） CRC32 校验，可校验 FLASH 中内核、RAMDISK 镜像文件是否完好。

（7） 设备驱动。串口、SDRAM、FLASH、以太网、LCD、NVRAM、EEPROM、键盘、USB、PCMCIA、PCI、RTC 等驱动支持。

（8） 上电自检功能。SDRAM、FLASH 大小自动检测；SDRAM 故障检测；CPU 型号。

（9） 特殊功能。XIP 内核引导。

4.4 BIOS BootLoader 配置方法

在前文介绍了 DOS+LoadCEPC 的启动方式。以这种方式启动 WINCE，最直接的好处就是不需要开发 BootLoader 程序，而且简单易用。但也有它的不足之处，因为系统启动是先执行的 DOS 程序，这样 DOS 系统存在的某些风险会间接影响 WINCE 系统在启动之前的稳定性和可靠性。

实际上，微软还提供了另一种启动方式，微软的官方文档称之为“BIOS Boot Loader”。实际上，称之为“NTLoader”更加合适一些。它的启动方式类似于 Windows2000 的启动方式。通过修改硬盘的“Boot Section”，再利用一个“BLDR”的文件来实现启动。

虽然微软很早就有“BIOS Boot Loader”，但真正“好用”的版本直到 Windows Embedded CE 6.0 R2 才出现，之前的版本因为对文件系统支持不佳，实为鸡肋一个，并不实用。所以，如果真正想使用“BIOS Boot Loader”，最好还是寻找 Windows Embedded CE 6.0 R2 版本。制作“BIOS Boot Loader”的工具是包含在 Windows Embedded CE 6.0 R2 版本中的一组独立文件，复制出来之后可以单独使用。并且，它支持 WINCE6.0 以及 WINCE5.0 系统启动。

以下详述制作 “BIOS Boot Loader”的过程。

如图 4-3 所示，在目录“C:\WINCE600\PLATFORM\CEPC\SRC\BOOTLOADER\BIOSLOADER\DISKIMAGES\SETUPDISK”下有制作“BIOS Boot Loader”的相关工具。首先制作一张 DOS 启动软盘或者 DOS 启动光盘，然后将该文件夹下的所有文件都复制到软盘或者光盘上。利用本目录下的工具软件“fdisk.exe”和“format.exe”对系统进行分区和格式化。注意不要使用其他的工具软件分区、格式化，否则可能会导致“BIOS Boot Loader”制作失败。

最后执行本目录下的“mkdisk.bat”，该批处理将自动建立好“Boot Section”与“BLDR”

文件。

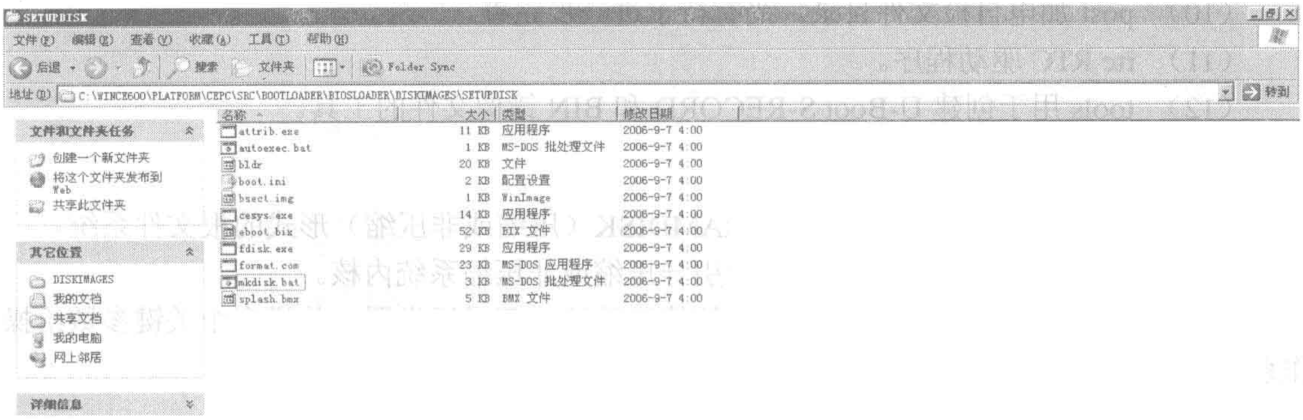


图 4-3

“BIOS Boot Loader”的另一种使用方法是将“BLDR”文件直接刷写 BIOS，变成真正的“BIOS Boot”。理论上说，这是完全可行的，但鉴于风险太高，笔者未曾测试。

如图 4-4 所示，是“BIOS Boot Loader”执行效果。

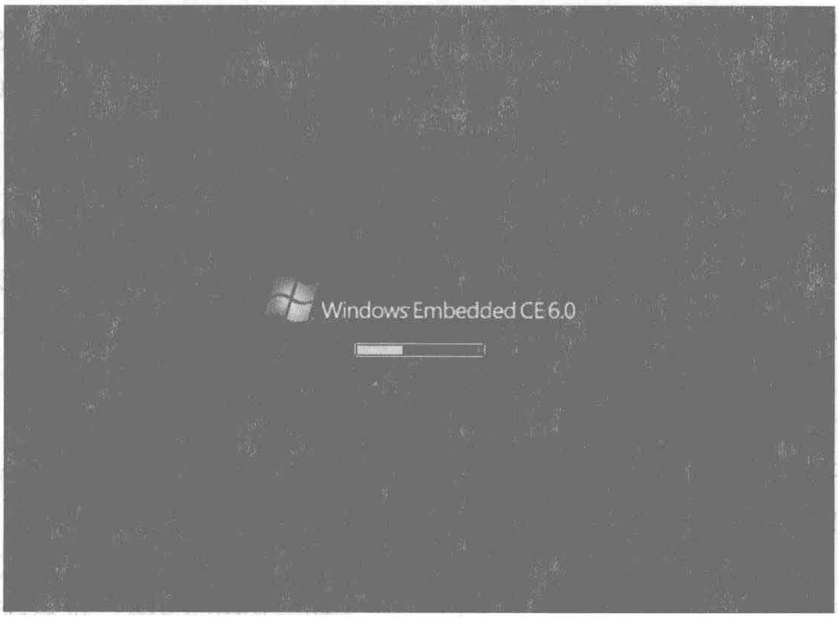


图 4-4

第5章 文件系统

5.1 Windows CE 文件系统体系结构

Windows CE 文件系统采用灵活的模块化设计，它允许自定义文件系统、筛选器和多种不同的块设备类型。文件系统和所有与文件相关的 API 都是通过 FileSys.exe 进程来管理的。这个模块实现了对象存储和存储管理器（我们将稍微讨论一下对象存储），并将所有文件系统统一到一个根目录“\”下面的单个系统中。在 Windows CE 中，所有文件和文件系统都存在于从反斜杠“\”作为根目录开始的单个命名空间中。所有文件均以在层次结构树中从根开始的唯一路径进行标识。这类似于桌面计算机版本的 Windows，只是没有驱动器号。在 Windows CE 中，驱动器作为文件夹装入根的下面。因此，添加到系统中的新存储卡将装入树的根中，其路径类似于“\Storage Card”。

FileSys.exe 由下列几个组件组成：

- ROM 文件系统
- 存储管理器
- 对象存储

对象存储是一个内存堆，由 FileSys.exe 控制。对象存储包含 RAM 系统注册表、RAM 文件系统和属性数据库。它们都是 FileSys.exe 模块的可选组件。RAM 文件系统和属性数据库是完全可选的，并且在某些系统中可以根本不存在。对每个 Windows CE 设备来说，以某些形式存在的注册表是必需的。直到 Windows CE 的 4.0 版本，它总是驻留在对象存储中。使用 Windows CE .NET，它可以作为文件存在于外部装入的文件系统（例如磁盘）中。

基于 RAM 的文件系统通常连接到呈现给应用程序的统一文件系统的根。就是说，文件“\MyFile.txt”位于统一系统的根和 RAM 文件系统的根中。ROM 文件系统连接到统一文件系统中的“\Windows”文件夹。这意味着，ROM 中的所有文件均可作为“\Windows”文件夹中的只读文件来访问。

存储管理器（Storage Manager）是 Windows CE 的新功能。如名称所示，它负责管理系统中的存储设备，以及用于访问它们的文件系统。存储管理器处理 4 种主要项目：

存储驱动程序。它们是物理存储介质的设备驱动程序。它们有时称为“块驱动程序”，这是因为它们提供对数据存储的随机寻址块的访问。

分区驱动程序。它们为单个存储设备上的多个分区提供管理。Windows CE 允许物理磁盘包含多个分区，并且每个分区可以格式化为不同的文件系统。分区驱动程序实际上是存储驱动程序的转换器。它公开与存储驱动程序相同的接口，并将分区的块地址转换为存储设备块的真实地址。然后，它将调用传递给存储驱动程序。

文件系统驱动程序。这些驱动程序将存储设备上的数据组织为文件和文件夹。Windows CE 附带了几个不同的系统，包括用于 CD 和 DVD 的 UDFS，以及 FATFS（包括 FAT32 支持）。

文件系统筛选器。文件系统筛选器用于处理对文件系统的调用，此后，文件系统才能获得这些调用。这就允许对文件访问进行某些特殊的处理，以便进行数据加密、压缩和使用统计数据进行监视。

图 5-1 说明了文件系统的各个组件之间的关系。

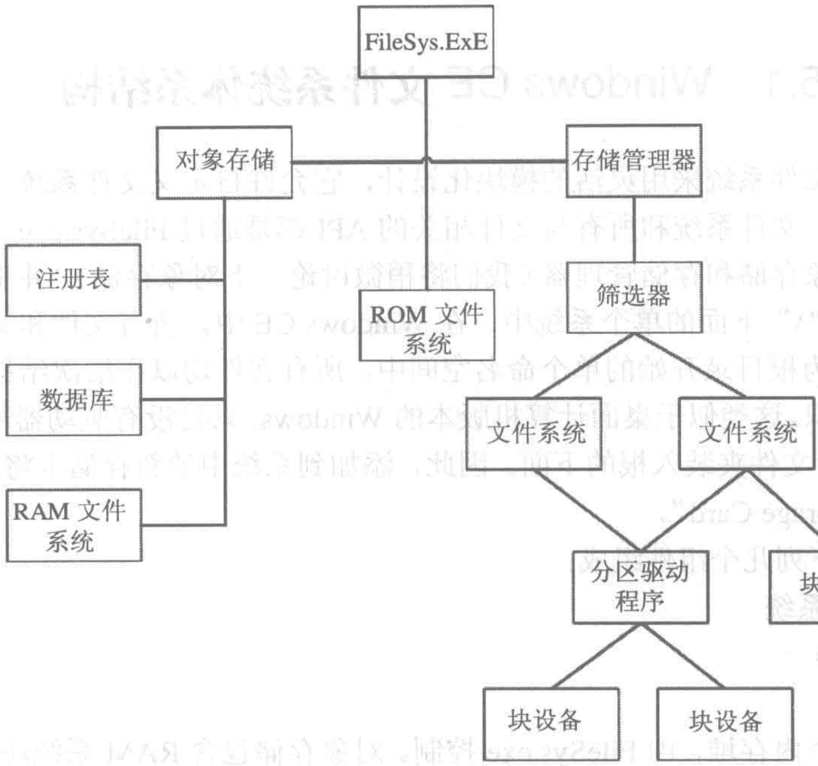


图 5-1

有关该结构，一个要注意到的重要事情是，文件系统筛选器工作于存储管理器的下面，并且无法应用于对象存储中的 ROM 文件系统或 RAM 文件系统。此时，Microsoft 没有为筛选对这些系统的访问提供机制。因此，在本文中我们将重点讨论上图右侧的内容，如图 5-2 所示。

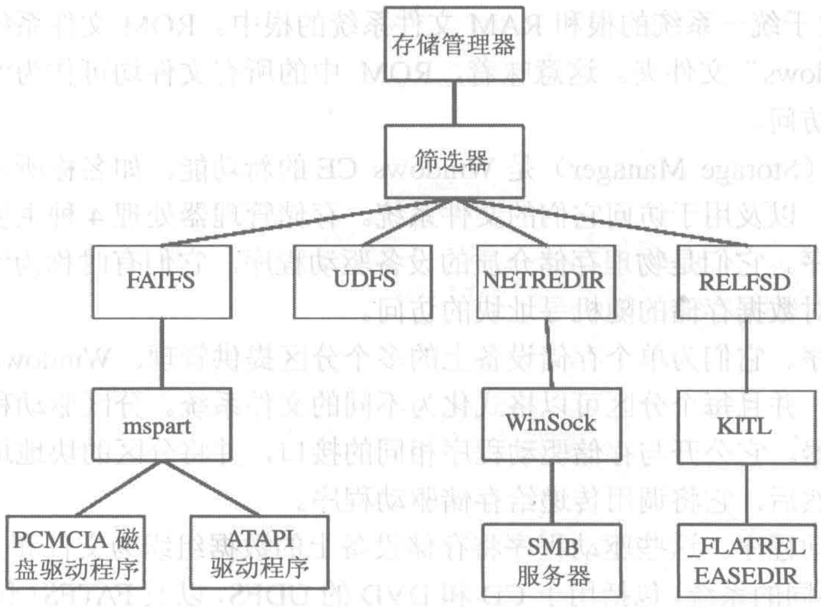


图 5-2

在上图中可以看见，并非所有文件系统驱动程序都使用了物理设备，即使使用，也可能没有使用分区驱动程序。这就提供了巨大的灵活性。例如，负责提供网络共享访问的网络重定向器使用 WinSock 通过网络与远程服务器通信，并且它在 Windows CE 设备上没有物理磁盘。

既然我们可以看见大多数项目是什么，以及它们是如何相互关联的，下面我们将讨论系统如何加载所有各项。操作系统启动时，NK.exe 将直接从 ROM 文件系统加载 FileSys.exe。然后，FileSys.exe 从 ROM 文件系统内的默认注册表对注册表进行初始化。(这里，在使用配置单元注册表时有一个类似“先有鸡还是先有蛋”的问题，这是因为注册表在磁盘上的文件中，而文件系统还没有装入。在随后的文章介绍配置单元注册表时，我们将介绍操作系统是如何解决这个问题的。)

然后，FileSys.exe 将读取注册表项，以便启动各种应用程序。首先列在注册表中的一个应用程序通常是 Device.exe，即设备管理器。设备管理器从 HKEY_LOCAL_MACHINE\Driver\BuiltIn 项加载驱动程序。正常情况下，任何内置的磁盘设备（例如，硬盘）列在该项下面，所以将加载块驱动程序。块驱动程序通告一个特定的设备类标识符 BLOCK_DRIVER_GUID {A4E7EDDA-E575-4252-9D6B-4195D48BB865}。

内置到 FileSys.exe 中的存储管理器向设备管理器通知系统注册，以便接收有关块驱动程序加载和卸载的通知。然后，存储管理器打开块驱动程序，并向它查询配置文件名称。每个块设备类型都有一个与它相关的配置文件。PROFILE 是一个注册表项，用于指定特定类型设备的分区驱动程序和默认文件系统（我们将对配置文件的注册表项的细节稍加介绍）。

存储管理器读取有关设备的分区驱动程序的信息，并加载适当的驱动程序（Microsoft 提供了一个称为“mspar”的分区驱动程序，用于通过磁盘的主启动记录中的分区表进行标准硬盘分区。当然，如果需要，用户可以随便创建自己的分区，也可以根本不使用它）。

一旦分区驱动程序已加载，然后存储管理器将请求分区驱动程序枚举磁盘上的分区，并标识每个分区上的文件系统。分区驱动程序将从主启动记录（MBR）中读取有关分区和文件系统的信息，并向存储管理器提供信息，然后，存储管理器使用该信息来加载每个分区的文件系统驱动程序，并将文件系统装入到统一文件系统的根中。虽然这似乎有很多步骤，但它允许完全在同一个框架中灵活地支持网络重定向器 FATFS 和 DVD ROM。

在了解 FileSys.exe 如何加载各种组件的步骤以后，我们将更为详细地介绍文件系统驱动程序和文件系统驱动程序管理器（FSDMGR）的角色。FSDMGR 是存储管理器（在该操作系统以前的版本中它是设备管理器的一部分）的一部分，负责向文件系统驱动程序提供服务。因为文件系统将不需要知道数据是否来自磁盘上的分区或者直接来自磁盘，所以，FSDMGR 对文件系统驱动程序进行包装，以便为驱动程序的高端或低端提供接口。图 5-3 说明这是如何工作的。

图 5-3

存储管理器调用文件系统驱动程序(FSD)，而 FSD 使用 FSDMGR_API 从设备检索数据。

如果是 CD（没有分区），则设备通过 FSDMGR 与块驱动程序通信。如果它是有多个分区的硬盘，那么它以同样方式使用 FSDMGR_API。但这之后 FSDMGR 会将工作转交给适当的分区驱动程序。

我们已经讨论了存储管理器、FSDMGR、FSD、分区驱动程序和块驱动程序如何交互和互操作。让我们回过来详细讨论它们是如何加载的，并考察注册表中的配置文件的细节。前面已经提到过，配置文件只是一组注册表值，用于定义有关块设备和应当如何在系统中使用它的信息。配置文件位于：HKEY_LOCAL_MACHINE\System\StorageManager\Profiles 项的下面。

每个配置文件都是位于基本配置文件项的下面的，以此配置文件名称标识的项。例如，如果 Windows CE 设备上有一个硬盘，并且它确实使用硬盘配置文件，则配置文件位于 HKEY_LOCAL_MACHINE\System\StorageManager\Profiles\Hard Disk 下面。所有配置文件信息都包含在该配置文件项下面的命名值中。表 5-1 列出了各种值及其说明。

表 5-1

值	类 型	说 明
Folder	REG_SZ	在 Windows 资源管理器中显示给用户的文件夹名称。对于多个实例，将自动追加整数（例如，Storage Card、Storage Card2 等等）
FileSystem	REG_SZ	用作磁盘的默认文件系统的名称（如果使用了分区驱动程序，则通常不使用它）
PartitionDriver	REG_SZ	列出如果默认驱动程序不合适时要使用的分区驱动程序。如果该字符串为空，则不加载任何分区驱动程序。如果该值不出现，则使用默认分区驱动程序
AutoFormat	REG_DWORD	如果磁盘没有格式化，则自动执行格式化
AutoPart	REG_DWORD	如果磁盘没有分区，则自动将它分区，并且其中一个分区占据最大数量的可用磁盘空间
AutoMount	REG_DWORD	当存储设备驱动程序加载时，自动装入文件系统
Name	REG_SZ	显示在控制面板 UI 中的配置文件名称
MountFlags	REG_DWORD	用于确定如何装入文件系统的标志（详细信息，请参阅下表）

需要特别注意 MountFlags 值，它是表 5-2 中的值的位掩码。

表 5-2

标 志	说 明
①	隐藏文件系统
②	可以包含配置单元注册表
④	作为文件系统的根（"\"）装入
⑧	隐藏 ROM 文件系统。（只与[4]一起使用。）

将文件系统标记为隐藏，可以防止它被任何标准的文件和文件夹枚举发现（例如 FindFirstFile 等等）。存储管理器独立完成该操作，以便设备驱动程序和应用程序可以检测到是否特定系统正在使用存储管理器（由于较旧版本的操作系统没有它，所以某些驱动程序可能需要与旧的 LoadFSD（Ex）机制相兼容，以用于加载文件系统）。虽然无法使用 FindFirstFile 来枚举任意的隐藏系统，但如果知道文件系统的名称，则可以在任何会使用文件路径的地方使用它。以下示例显示了如何检测存储管理器是否正在使用某个系统。

```
BOOL IsStorageManagerRunning()
{
```

```
    DWORD attr = GetFileAttributes( _T("\\StoreMgr") );
    if( (attr != -1) && (attr & FILE_ATTRIBUTE_TEMPORARY) )
        return TRUE;
    return FALSE;
}
```

MountFlags 的下一位指示文件系统是否可包含基于配置单元的注册表。这使 FileSys.exe 能够管理先前提到的“先有鸡还是先有蛋”的问题（因为需要注册表才能加载访问可能在磁盘上的注册表配置单元文件所需的组件……）我们将在随后的有关配置单元注册表的文章中介绍如何使用这一位。

后面两位是相关的，当用户希望将外部文件系统作为统一文件系统的根装入时，则需要使用它们。可以回想起，通常统一系统的根是 RAM 文件系统。这对电池供电的手持设备很有效，但对有时会关闭的交流电源供电的设备却无效，因为 RAM 内容每次关闭时都会丢失。作为根标志的装入文件系统允许您通过将外部存储作为根进行连接来避免这个问题，因为这样一来，文件\MyDataFile.TXT 将驻留在外部存储设备的根中。隐藏 ROM 文件系统将隐藏 ROM 文件系统数据文件（但不执行适当的 EXE 和 DLL），以便允许用户更新 ROM 中的所有文件。这就允许用户在闪存中使用非常小的操作系统映像，而将大多数可执行文件放在磁盘上，只在需要时才加载它们（现在很像桌面计算机系统）。

如果特定配置文件的任何值都不存在，则存储管理器将使用 HKEY_LOCAL_MACHINE\System\StorageManager\Profiles 项中的默认值。可以重写的默认值位于 COMMON.REG 中。用户应当使用您的 PLATFORM.REG 或 PROJECT.REG 进行重写。表 5-3 显示了 COMMON.REG 中的默认值。

表 5-3

值	默认值
Folder	LOC_STORE_DEFAULT_FOLDER（用于设备的.STR 文件中一个字符串的标识符；在英文内部版本中通常是“Storage Card”）
FileSystem	FATFS
PartitionDriver	Mspart.dll
AutoFormat	0
AutoPart	0
AutoMount	1
MountFlags	0

5.2 实现 Windows CE 6 的硬盘文件系统

WINCE 上所使用的文件系统一般都为 FAT 文件系统，但在 WINCE6 中新增加了 exFAT 文件系统以及 TFAT 文件系统。

WINCE6 自带的 X86 平台的 BSP，提供了上层 FAT 文件系统的完整驱动，要想在操作 WINCE 操作系统中正确识别硬盘，以下几个组件最好加入系统中，如图 5-4 所示。

SYSGEN_STOREMGR

SYSGEN_FATFS

SYSGEN_MSPART

SYSGEN_STOREMGR_CPL

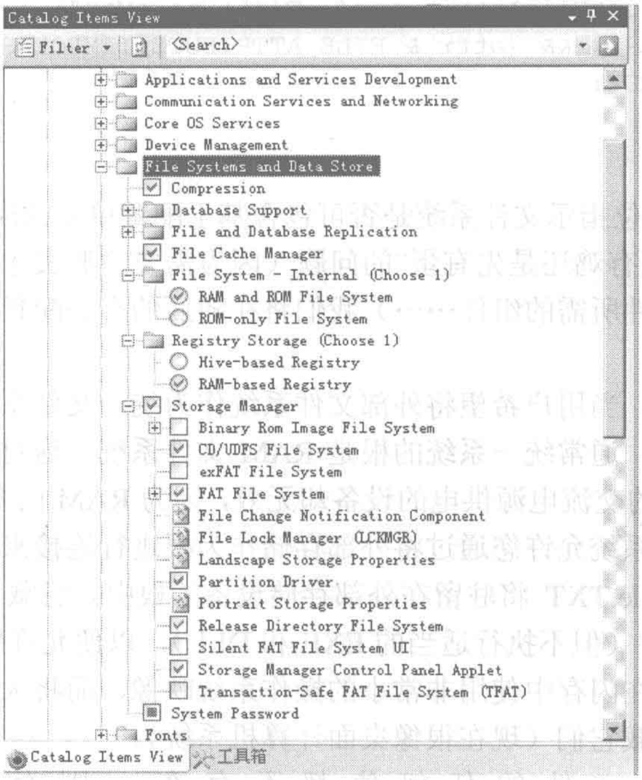


图 5-4

上述文件系统上层驱动的源代码位于“C:\WINCE600\PRIVATE\WINCEOS\COREOS\STORAGE”文件夹下。

其中，最常用的 FAT 文件系统驱动位于“C:\WINCE600\PRIVATE\WINCEOS\COREOS\STORAGE\DOSPART”文件夹下。

而最复杂的“存储管理器”FSDMGR 的源码位于“C:\WINCE600\PRIVATE\WINCEOS\COREOS\STORAGE\FSDMGR”文件夹下。

另外，WINCE6 中新增加了 exFAT 文件系统。exFAT，全称 Extended File Allocation Table File System，即扩展 FAT，也就是扩展文件分配表，是 Microsoft 在 WINCE6.0（包括 Windows CE 6.0、Windows Mobile）中引入的一种适合于闪存的文件系统。对于闪存，NTFS 文件系统过于复杂，exFAT 更为适用。

相对 FAT 文件系统，exFAT 有如下好处：

- 增强了台式电脑与移动设备的互操作能力；
- 没有文件大小 4 GB 的限制；
- 簇大小可高达 32 MB；
- 采用了剩余空间分配表，剩余空间分配性能改进；
- 同一目录下最大文件数超过 1000 个；
- 支持访问控制；
- 支持 TFAT。

采用该文件系统的闪存盘不支持 Windows Vista ReadyBoost。Windows Vista SP1 支持该文件系统。

但是，exFAT 只是一个折中的方案，只为 U 盘而生。

现在超过 4 GB 的 U 盘格式化时默认是 NTFS 分区，但是这种格式是很伤硬盘的，因为 NTFS 分区是采用“日志式”的文件系统，需要记录详细的读写操作，肯定会比较伤闪盘芯片，因为要不断读写。

当然，光有上层文件系统的驱动还是不够的，还需要底层芯片驱动支持才能真正识别硬盘。这也是有些主板无论如何也不能识别硬盘的原因。

硬件芯片驱动需要加入的组件如下：

SYSGEN_ATAPI_PCIO

SYSGEN_ATAPI_PCIO_CD

SYSGEN_ATAPI

SYSGEN_ATAPI_PCIP_PDC20262

SYSGEN_ATADISK

实际上，这些组件兼容的芯片为 ICH4 之前的版本，之后的版本并没有在 WINCE6 中包含。当然，升级到 WINCE6 R2 版本，并且把补丁打上，会有新的芯片驱动程序支持。

具体芯片驱动的源代码位于“C:\WINCE600\PUBLIC\COMMON\OAK\DRIVERS\BLOCK\ATAPI”文件中。从源码文件可以看出，WINCE6 支持的硬件芯片型号有：ALI1453、pdc20262、i82371 等。

值得注意的是，如果目标主板上的硬盘驱动芯片不被 WINCE6 支持，并不代表目标板上，就不能使用 WINCE6 操作系统。实际上，只要 DOS 系统能够识别硬盘，那么 WINCE6 正常启动应该不成问题。WINCE 操作系统启动之后，如果要使用硬盘，当然，还可以采用外接 U 盘当成硬盘来使用，但我们一般不推荐这种使用方法。实际上，新的硬盘驱动芯片的驱动程序在各芯片组厂商的官方网站上一般都可以找到，将它们移植到自己的平台中也不复杂。

上述芯片驱动组件在 PLATFORM BUILDER 平台上的位置如图 5-5 所示。

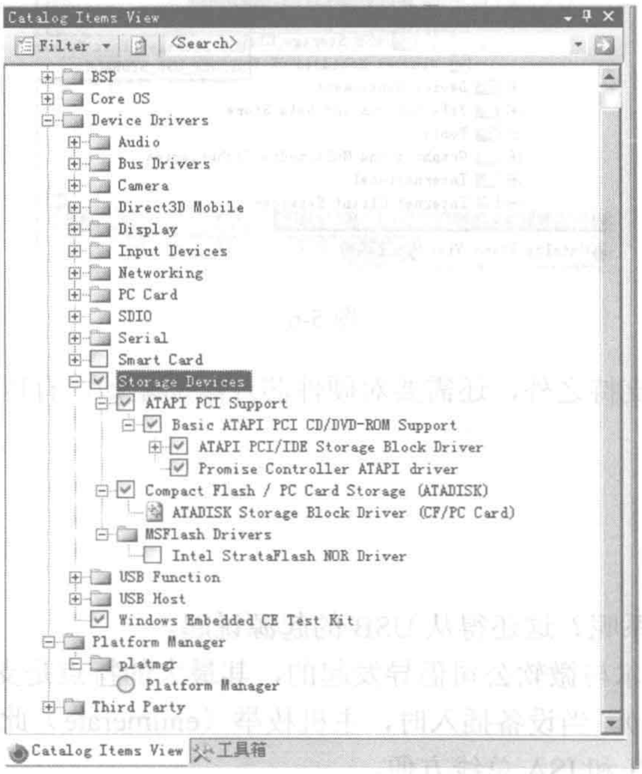


图 5-5

5.3 实现 Windows CE 6 的 USB 文件系统

所有的 USB 基本上都是通用协议，同具体芯片型号关系不大。因此，在 WINCE6 平台上，几乎没有不能识别的标准 USB 主从芯片。要想让 WINCE 操作系统识别 USB 硬盘设备，除了上节所述标准的 FAT 文件系统驱动需要加入外，以下系统组件也需要加入 NK.BIN 当中。

SYSGEN_USB_STORAGE

这一组件实际上是对支持 USB Mass Storage 协议的支持，该标准协议可以支持 USB 接口的移动硬盘、U 盘、光驱、刻录机等设备。在 PLATFORM BUILDER 平台上的位置如图 5-6 所示。

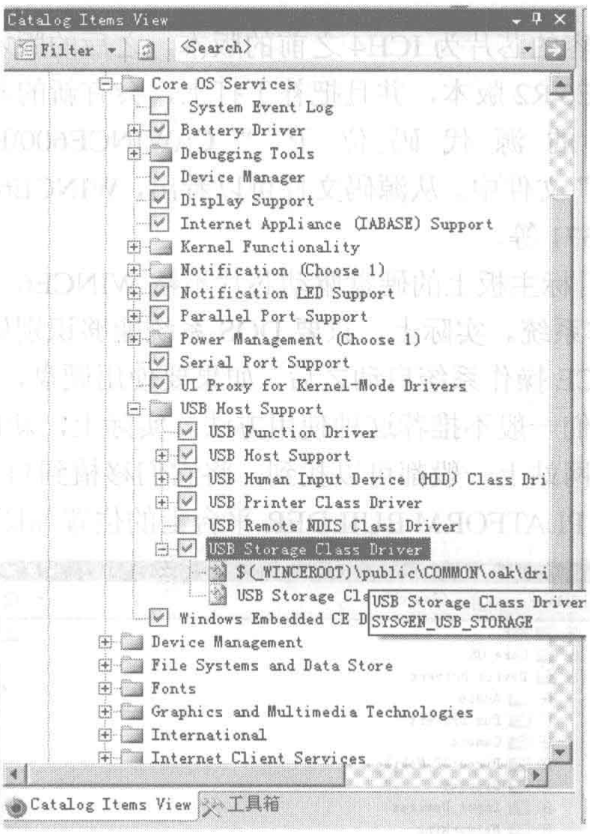


图 5-6

当然，除了对协议支持之外，还需要对硬件芯片添加驱动，有以下几个驱动组件可选，如图 5-7 所示。

BSP_USB_EHCI

BSP_USB_OHCI

BSP_USB_UHCI

这几个组件有何关系呢？这还得从 USB 的起源说起。

USB 最初是由英特尔与微软公司倡导发起的，其最大的特点是支持热插拔（Hot plug）和即插即用（Plug&Play）。当设备插入时，主机枚举（enumerate）此设备并加载所需的驱动程序，因此使用远比 PCI 和 ISA 总线方便。

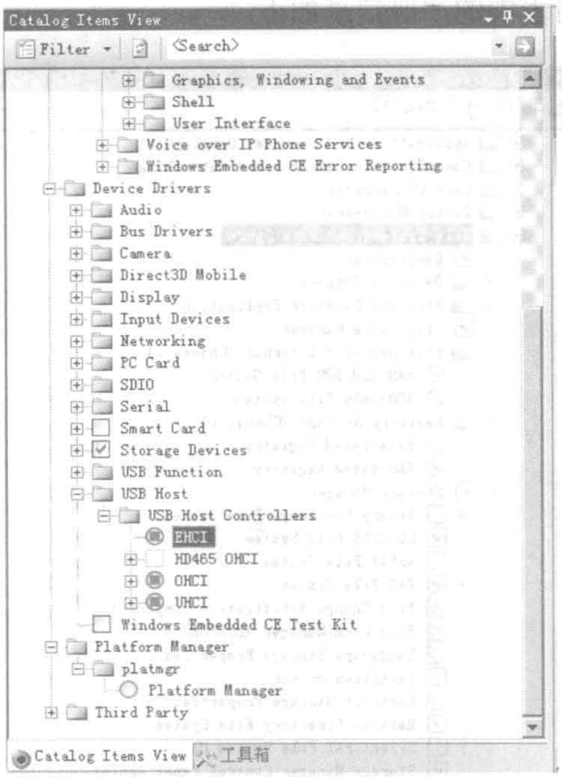


图 5-7

USB 标准由 USB Implementers Forum （USBIF）负责制定，其成员包括苹果电脑、惠普、NEC、Microsoft 和 Intel。USBIF 于 2001 年底公布了 2.0 规范，之前还有 0.9、1.0 和 1.1 版，它们都是完全向后兼容的。On-The-Go Supplement to the USB 2.0 Specification 的当前版本号是 1.0a。

USB1.0 和 USB1.1 的标准由两个竞争的 USB 主控制器实现，即康柏的开放主机控制器接口（OHCI）和 Intel 的通用主机控制器接口（UHCI）。VIA（威盛）采纳了 UHCI；其他主要的芯片组多使用 OHCI。它们的主要区别是，UHCI 更加依赖软件驱动，因此对 CPU 要求更高，但是自身的硬件会更廉价。它们的并存导致操作系统开发和硬件厂商都必须在两个方案上开发和测试，从而导致费用上升。因此 USBIF 在 USB 2.0 的设计阶段坚持只能有一个实现规范，这就是扩展主机控制器接口（EHCI）。因为 EHCI 只支持全速传输，所以 EHCI 控制器包括四个虚拟的全速或者慢速控制器。这样 Intel 和 VIA 都使用虚拟 UHCI，其他公司一般使用 OHCI 控制器。

因此，USB 主机端控制芯片的驱动一般都符合上述标准，也就为 USB 硬盘文件系统的实现清除了所有障碍。

通常，有些新的主板会出现 WINCE 无法识别硬盘的错误，但一般都不会出现无法识别 U 盘的错误。

5.4　实现 Windows CE 6 的光盘文件系统

CD/UDFS 文件系统，这两种文件系统被用于读取 CD、DVD 等。除了通过在 PB 的“catalog”中加入这个文件系统外，如图 5-8 所示，还可以在 PB 中添加 SYSGEN_UDFS 环境变量来实

现。CDFS 和 UDFS 在注册表中的注册信息如下：

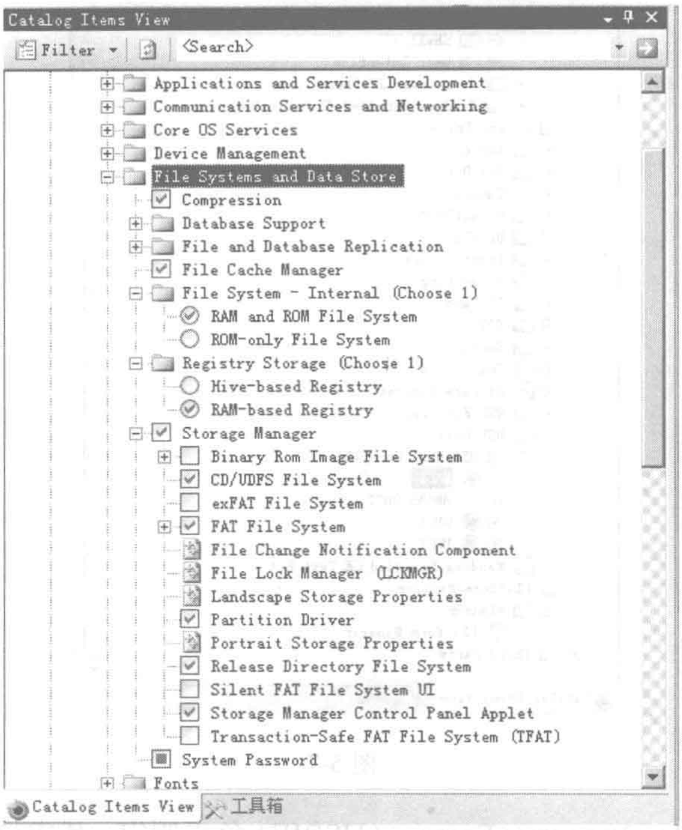


图 5-8

```
; Default values for udfs. These can be overridden per profile.
    [HKEY_LOCAL_MACHINE\System\StorageManager\UDFS]
    "FriendlyName"="CDFS/UDFS FileSystem"
    "Dll"="udfs.dll"
    "Paging"=dword:1
    [HKEY_LOCAL_MACHINE\System\StorageManager\Profiles\CDProfile]
    "Name"="IDE CDROM/DVD Drive"
    "Folder"="CDROM Drive"
    "DefaultFileSystem"="UDFS"
    "PartitionDriver"=""
```

上面注册表信息在文件 common.reg 中。注册表数据从上至下有效，也就是说下面的数据可以覆盖上面的数据。从注册表数据可以看出 udfs.dll 包含了 UDFS 文件系统的驱动程序，CDROM 的驱动器名为“CDROM Drive”，采用的文件系统为 UDFS，没有分区驱动程序。

第 6 章　USB 驱动

6.1　USB 驱动开发介绍

随着 USB 设备的普及，摆在开发人员面前的驱动开发任务也越来越繁重了，特别是对于一些嵌入式开发厂商来讲，由于设备所采用的操作系统不同，相应的硬件接口也是不一样的，开发相关的 USB 驱动程序更是难上加难。

下面简略介绍 Windows CE.NET 中 USB 设备驱动开发的一些基础知识。

Windows CE 的 USB 系统软件分为两层：USB Client 设备驱动程序和底层的 Windows CE 实现的函数层。USB 设备驱动程序主要负责利用系统提供的底层接口配置设备，和设备进行通信。底层的函数其本身又由两部分组成，通用串行总线驱动程序（USBD）模块和较低的主控制器驱动程序（HCD）模块。HCD 负责最底层的处理，USBD 模块实现较高的 USBD 函数接口。USB 设备驱动主要利用 USBD 接口函数和它们的外围设备打交道。

USB 设备驱动程序主要和 USBD 打交道，所以我们必须详细地了解 USBD 提供的函数。

主要的传输函数有：

```
AbourtTransfer        IssueControlTransfer
CloseTransfer         IssueInterrupTransfer
GetIsochResult        IssueIsochTransfer
GetTransferStatus     IstransferComplete
IssueBulkTransfer     IssueVendorTransfer
```

主要的用于打开和关闭 USBD 和 USB 设备之间的通信通道的函数有：

```
AbortPipeTransfers     ClosePipe
IsDefaultPipeHalted    IsPipeHalted
OpenPipe               ResetDefaultPipe
ResetPipe
```

相应的打包函数接口有：

```
GetFrameLength    GetFrameNumber    ReleaseFrameLengthControl
SetFrameLength    TakeFrameLengthControl
```

取得设置设备配置的函数有：

```
ClearFeature     SetDescriptor
GetDescriptor    SetFeature
GetInterface     SetInterface
GetStatus        SyncFrame
```

与 USB 进行交互的实现方法相关的多任务函数有：

```
FindInterface                    RegisterClientDeviceId
GetDeviceInfo                    RegisterClientSettings
GetUSBDVersion                   RegisterNotificationRoutine
LoadGenericInterfaceDriver       TranslateStringDescr
OpenClientRegisterKey            UnRegisterNotificationRoutine
```

常见的 Windows CE.NET 下 USB 的设备驱动程序的编写有以下几种方法。

- 流式接口函数

这种驱动程序主要呈现流式函数接口，主要输出 XXX_Init、XXX_Deinit、XXX_Open、XXX_Close、XXX_Read、XXX_Write、XXX_Seek、XXX_IOControl、XXX_PowerUp、XXX_PowerDown 等流式接口，注意上述的几个接口一定都要输出，另外 XXX 必须为三个字符，否则会出错。但是此类的驱动程序不是通过设备管理接口来加载的，所以必须手工调用 RegisterDevice()和 DeregisterDevice()函数来加载和卸载驱动程序。用户可以将此类的设备作为标准的文件来操作，只要调用相应的文件操作就可以和驱动程序打交道。

- 使用现有的 Windows CE.NET 的应用程序接口

此类设备主要是利用 Windows CE.NET 中已经有了现成的函数接口，例如 USB Mass Storage Disk，它主要利用现有的 Windows CE.Net 中已经有的可安装文件系统接口，呈现给系统可用的文件系统，对于用户来讲，它是透明的，用户仅仅感觉在操作一个文件夹。

- 创建指定到特定的 USBD 的用户指定的 API

这种方法在 USBD 呈现设备时不需要任何限制，主要是特制提供 API 给用户，一般不太常见。

USB 设备驱动程序必须输出的函数如下。

- USBDeviecAttach

当 USB 设备连接到计算机上时，USBD 模块就会调用此函数，这个函数主要用于初始化 USB 设备，取得 USB 设备信息，配置 USB 设备，并且申请必需的资源。

- USBInstallDriver

主要用于创建一个驱动程序加载所需的注册表信息，例如读写超时、设备名称等。

- USBUninstallDriver

主要用于释放驱动程序所占用的资源，以及删除 USBInstallDriver 函数创建的注册表等。

上述的三个函数接口是所有的 USB 驱动程序必须提供的，缺一不可。

另外比较重要的是 USB 设备驱动程序的注册表配置，一般的 USB 设备驱动程序的注册表配置在 HKEY_LOCAL_MACHINE\Drivers\USB\LoadClients 下，每个驱动程序的子键都有 Group1_ID\Group2_ID\Group3_ID\DriverName 格式，如果注册表信息与 USB 设备信息符合，USBD 就会加载此驱动程序。否则设备的子键应该由供应商、设备类和协议信息通过下画线组成。

具体的配置举例如下：

例如，用户有个 PDA 设备，它具有一个 USB 接口，它的供应厂商 ID 假设为 0x0888，设备 ID 为 0x0999，没有使用特殊的协议，那么它的加载注册表应该写为：

[HKEY_LOCAL_MACHINE\Drivers\USB\LoadClients\2184_2457\Default\Default\PDA]
"DLL"="pdausb.dll"

需要注意的是注册表构成都是十进制数值来标识的，注意一下十进制和十六进制的转换。

再举个 USB 鼠标的例子，USB 鼠标是标准的 HID 设备，它的协议：InterfaceClassCode 为 3（HID 类），InterfaceSubclassCode 为 1（引导接口类），InterfaceProtocolCode 为 2（鼠标协议类），所以它的注册如下：

[HKEY_LOCAL_MACHINE\Drivers\USB\LoadClients\Default\Default\3_1_2\USBMouse]
"DLL"="usbmouse.dll"

到此为止，我们可以看出，其实驱动开发无非做两件事情，一件是和硬件打交道，另外一件是和操作系统打交道。举个简单的例子，例如：我们需要开发一个 USB 鼠标驱动程序，我们就需要了解 USB 鼠标硬件上是怎么发送数据的？操作系统怎么才能得到鼠标的控制事件？其实 USB 鼠标是有一个中断 PIPE 的，用于传送鼠标产生的数据，Windwos CE.NET 中有个接口函数叫做 mouse_event(),专门用于产生鼠标事件，但它不关心具体是什么硬件，甚至我们自己在应用程序中调用这个函数都可以实现模拟鼠标，对应的有个 keybd_event()，用于产生键盘事件，知道了这个就好办多了，只要将相应的数据转换一下，调用一下 mouse_event() 即可，下面举个简单的例子来详细说明驱动程序的开发过程。

例如有个 USB Mouse 设备，设备信息描述如下：

```
Device Descriptor:
bcdUSB: 0x0100
bDeviceClass: 0x00
bDeviceSubClass: 0x00
bDeviceProtocol: 0x00
bMaxPacketSize0: 0x08 (8)
idVendor: 0x05E3 (Genesys Logic Inc.)
idProduct: 0x0001
bcdDevice: 0x0101
iManufacturer: 0x00
iProduct: 0x01
iSerialNumber: 0x00
bNumConfigurations: 0x01

ConnectionStatus: DeviceConnected
Current Config Value: 0x01
Device Bus Speed: Low
Device Address: 0x02
Open Pipes: 1

Endpoint Descriptor:
bEndpointAddress: 0x81
Transfer Type: Interrupt
wMaxPacketSize: 0x0003 (3)
bInterval: 0x0A
```

可以看出上述设备有一个中断 PIPE，包的最大值为 3。可能有人问上述的值怎么得到的，win2k 的 DDK 中有个 usbview 的例程，编译一下，将你的 USB 设备插到 PC 机的 USB 口中，运行 usbview.exe 即可看得相应的设备信息。

有了这些基本信息，就可以编写 USB 设备了。首先声明一下，下面的代码取自微软的 USB 鼠标样本程序，版权归微软所有，此处仅仅借用来描述 USB 鼠标驱动的开发过程，读者如需要引用此代码，需要得到微软的同意。

首先，必须输出 USBD 要求调用的三个函数，当设备插入到 USB 端口时，USBD 会调用 USBDeviceAttach()函数，相应的代码如下：

```
extern "C" BOOL
USBDeviceAttach(
USB_HANDLE hDevice, // USB设备句柄
LPCUSB_FUNCS lpUsbFuncs, // USBDI的函数集合
LPCUSB_INTERFACE lpInterface, // 设备接口描述信息
LPCWSTR szUniqueDriverId, // 设备ID描述字符串。
LPBOOL fAcceptControl, // 返回TRUE，标识我们可以控制此设备，反之表示不能控制
DWORD dwUnused)
{
*fAcceptControl = FALSE;
// 我们的鼠标设备有特定的描述信息，要检测是否是我们的设备。
if (lpInterface == NULL)
return FALSE;
// 打印相关的USB设备接口描述信息。
DEBUGMSG(ZONE_INIT,(TEXT("USBMouse: DeviceAttach, IF %u, #EP:%u, Class:%u,
Sub:%u,Prot:%u\r\n"),
lpInterface->Descriptor.bInterfaceNumber,lpInterface->Descriptor.bNumEndpoints,
lpInterface->Descriptor.bInterfaceClass,lpInterface->Descriptor.bInterfaceSubC
lass,lpInterface->Descriptor.bInterfaceProtocol));
// 初试数据USB鼠标类，产生一个接受USB鼠标数据的线程
CMouse * pMouse = new CMouse(hDevice, lpUsbFuncs, lpInterface);
if (pMouse == NULL)
return FALSE;

if (!pMouse->Initialize())
{
delete pMouse;
return FALSE;
}
// 注册一个监控USB设备事件的回调函数，用于监控USB设备是否已经拔掉。
(*lpUsbFuncs->lpRegisterNotificationRoutine)(hDevice,
USBDeviceNotifications, pMouse);
*fAcceptControl = TRUE;
return TRUE;
}
```

第二个函数是 USBInstallDriver()函数，一些基本定义如下：

```
const WCHAR gcszRegisterClientDriverId[] = L"RegisterClientDriverID";
const WCHAR gcszRegisterClientSettings[] = L"RegisterClientSettings";
const WCHAR gcszUnRegisterClientDriverId[] = L"UnRegisterClientDriverID";
const WCHAR gcszUnRegisterClientSettings[] = L"UnRegisterClientSettings";
const WCHAR gcszMouseDriverId[] = L"Generic_Sample_Mouse_Driver";
函数接口如下:
extern "C" BOOL
USBInstallDriver(
LPCWSTR szDriverLibFile) // @parm [IN] - Contains client driver DLL name
{
BOOL fRet = FALSE;
```

```
HINSTANCE hInst = LoadLibrary(L"USBD.DLL");
// 注册USB设备信息
if(hInst)
{
LPREGISTER_CLIENT_DRIVER_ID pRegisterId = (LPREGISTER_CLIENT_DRIVER_ID)
GetProcAddress(hInst, gcszRegisterClientDriverId);

LPREGISTER_CLIENT_SETTINGS pRegisterSettings =
(LPREGISTER_CLIENT_SETTINGS) GetProcAddress(hInst,
gcszRegisterClientSettings);

if(pRegisterId && pRegisterSettings)
{
USB_DRIVER_SETTINGS DriverSettings;
DriverSettings.dwCount = sizeof(DriverSettings);

// 设置我们的特定的信息。
DriverSettings.dwVendorId = USB_NO_INFO;
DriverSettings.dwProductId = USB_NO_INFO;
DriverSettings.dwReleaseNumber = USB_NO_INFO;

DriverSettings.dwDeviceClass = USB_NO_INFO;
DriverSettings.dwDeviceSubClass = USB_NO_INFO;
DriverSettings.dwDeviceProtocol = USB_NO_INFO;

DriverSettings.dwInterfaceClass = 0x03; // HID
DriverSettings.dwInterfaceSubClass = 0x01; // boot device
DriverSettings.dwInterfaceProtocol = 0x02; // mouse

fRet = (*pRegisterId)(gcszMouseDriverId);

if(fRet)
{
fRet = (*pRegisterSettings)(szDriverLibFile,
gcszMouseDriverId, NULL, &DriverSettings);

if(!fRet)
{
//BUGBUG unregister the Client Driver's ID
}
}
}
else
{
RETAILMSG(1,(TEXT("!USBMouse: Error getting USBD function pointers\r\n")));
}
FreeLibrary(hInst);
}
return fRet;
}
```

上述代码主要用于产生 USB 设备驱动程序需要的注册表信息，需要注意的是：USB 设

备驱动程序不使用标准的注册表函数，而是使用 RegisterClientDriverID() 和 RegisterClientSettings 来注册相应的设备信息。

另外一个函数是 USBUninstallDriver()函数，具体代码如下：

```
extern "C" BOOL
USBUnInstallDriver()
{
BOOL fRet = FALSE;
HINSTANCE hInst = LoadLibrary(L"USBD.DLL");

if(hInst)
{
LPUN_REGISTER_CLIENT_DRIVER_ID pUnRegisterId =
(LPUN_REGISTER_CLIENT_DRIVER_ID)
GetProcAddress(hInst, gcszUnRegisterClientDriverId);

LPUN_REGISTER_CLIENT_SETTINGS pUnRegisterSettings =
(LPUN_REGISTER_CLIENT_SETTINGS) GetProcAddress(hInst,
gcszUnRegisterClientSettings);

if(pUnRegisterSettings)
{
USB_DRIVER_SETTINGS DriverSettings;

DriverSettings.dwCount = sizeof(DriverSettings);
// 必须填入与注册时相同的信息。
DriverSettings.dwVendorId = USB_NO_INFO;
DriverSettings.dwProductId = USB_NO_INFO;
DriverSettings.dwReleaseNumber = USB_NO_INFO;

DriverSettings.dwDeviceClass = USB_NO_INFO;
DriverSettings.dwDeviceSubClass = USB_NO_INFO;
DriverSettings.dwDeviceProtocol = USB_NO_INFO;

DriverSettings.dwInterfaceClass = 0x03; // HID
DriverSettings.dwInterfaceSubClass = 0x01; // boot device
DriverSettings.dwInterfaceProtocol = 0x02; // mouse

fRet = (*pUnRegisterSettings)(gcszMouseDriverId, NULL,
&DriverSettings);
}

if(pUnRegisterId)
{
BOOL fRetTemp = (*pUnRegisterId)(gcszMouseDriverId);
fRet = fRet ? fRetTemp : fRet;
}
FreeLibrary(hInst);
}
return fRet;
}
```

此函数主要用于删除 USBInstallDriver()时创建的注册表信息，同样地，它使用自己的函数接口 UnRegisterClientDriverID()和 UnRegisterClientSettings()来做相应的处理。

另外一个需要处理的注册的监控通知函数为 USBDeviceNotifications():

```
extern "C" BOOL USBDeviceNotifications(LPVOID lpvNotifyParameter, DWORD
dwCode,
LPDWORD * dwInfo1, LPDWORD * dwInfo2, LPDWORD * dwInfo3,
LPDWORD * dwInfo4)
{
CMouse * pMouse = (CMouse *)lpvNotifyParameter;

switch(dwCode)
{
case USB_CLOSE_DEVICE:
//删除相关的资源。
delete pMouse;
return TRUE;
}
return FALSE;
}
```

USB 鼠标的类的定义如下：

```
class CMouse
{
public:
CMouse::CMouse(USB_HANDLE hDevice, LPCUSB_FUNCS lpUsbFuncs,
LPCUSB_INTERFACE lpInterface);
~CMouse();

BOOL Initialize();
private:
// 传输完毕调用的回调函数
static DWORD CALLBACK MouseTransferCompleteStub(LPVOID lpvNotifyParameter);
// 中断处理函数
static ULONG CALLBACK CMouse::MouseThreadStub(PVOID context);
DWORD MouseTransferComplete();
DWORD MouseThread();

BOOL SubmitInterrupt();
BOOL HandleInterrupt();

BOOL m_fClosing;
BOOL m_fReadyForMouseEvents;

HANDLE m_hEvent;
HANDLE m_hThread;

USB_HANDLE m_hDevice;
USB_PIPE m_hInterruptPipe;
USB_TRANSFER m_hInterruptTransfer;
```

```
LPCUSB_FUNCS m_lpUsbFuncs;
LPCUSB_INTERFACE m_pInterface;

BOOL m_fPrevButton1;
BOOL m_fPrevButton2;
BOOL m_fPrevButton3;

// 数据接受缓冲区。
BYTE m_pbDataBuffer[8];
};
```

具体实现如下：

```
// 构造函数，初始化时调用
CMouse::CMouse(USB_HANDLE hDevice, LPCUSB_FUNCS lpUsbFuncs,
LPCUSB_INTERFACE lpInterface)
{
m_fClosing = FALSE;
m_fReadyForMouseEvents = FALSE;
m_hEvent = NULL;
m_hThread = NULL;

m_hDevice = hDevice;
m_hInterruptPipe = NULL;
m_hInterruptTransfer = NULL;

m_lpUsbFuncs = lpUsbFuncs;
m_pInterface = lpInterface;

m_fPrevButton1 = FALSE;
m_fPrevButton2 = FALSE;
m_fPrevButton3 = FALSE;

memset(m_pbDataBuffer, 0, sizeof(m_pbDataBuffer));
}

// 析构函数，用于清除申请的资源。
CMouse::~CMouse()
{
// 通知系统去关闭相关的函数接口。
m_fClosing = TRUE;

// Wake up the connection thread again and give it time to die.
if (m_hEvent != NULL)
{
// 通知关闭数据接受线程。
SetEvent(m_hEvent);

if (m_hThread != NULL)
{
```

```
DWORD dwWaitReturn;

dwWaitReturn = WaitForSingleObject(m_hThread, 1000);
if (dwWaitReturn != WAIT_OBJECT_0)
{
TerminateThread(m_hThread, DWORD(-1));
}
CloseHandle(m_hThread);
m_hThread = NULL;
}
CloseHandle(m_hEvent);
m_hEvent = NULL;
}

if(m_hInterruptTransfer)
(*m_lpUsbFuncs->lpCloseTransfer)(m_hInterruptTransfer);

if(m_hInterruptPipe)
(*m_lpUsbFuncs->lpClosePipe)(m_hInterruptPipe);
}

// 初始化USB鼠标驱动程序
BOOL CMouse::Initialize()
{
LPCUSB_DEVICE lpDeviceInfo = (*m_lpUsbFuncs->lpGetDeviceInfo)(m_hDevice);

// 检测配置：USB鼠标应该只有一个中断管道
if ((m_pInterface->lpEndpoints[0].Descriptor.bmAttributes &
USB_ENDPOINT_TYPE_MASK) != USB_ENDPOINT_TYPE_INTERRUPT)
{
RETAILMSG(1,(TEXT("!USBMouse: EP 0 wrong type (%u)!\r\n"),
m_pInterface->lpEndpoints[0].Descriptor.bmAttributes));
return FALSE;
}
DEBUGMSG(ZONE_INIT,(TEXT("USBMouse: EP 0:MaxPacket: %u, Interval: %u\r\n"),
m_pInterface->lpEndpoints[0].Descriptor.wMaxPacketSize,
m_pInterface->lpEndpoints[0].Descriptor.bInterval));

m_hInterruptPipe = (*m_lpUsbFuncs->lpOpenPipe)(m_hDevice,
&m_pInterface->lpEndpoints[0].Descriptor);

if (m_hInterruptPipe == NULL) {
RETAILMSG(1,(TEXT("Mouse: Error opening interrupt pipe\r\n")));
return (FALSE);
}
m_hEvent = CreateEvent(NULL, FALSE, FALSE, NULL);
if (m_hEvent == NULL)
{
RETAILMSG(1,(TEXT("USBMouse: Error on CreateEvent for connect event\r\n")));
return(FALSE);
}
```

```
// 创建数据接受线程
m_hThread = CreateThread(0, 0, MouseThreadStub, this, 0, NULL);
if (m_hThread == NULL)
{
RETAILMSG(1,(TEXT("USBMouse: Error on CreateThread\r\n")));
return(FALSE);
}

return(TRUE);
}

// 从USB鼠标设备中读出数据，产生相应的鼠标事件。
BOOL CMouse::SubmitInterrupt()
{
if(m_hInterruptTransfer)
(*m_lpUsbFuncs->lpCloseTransfer)(m_hInterruptTransfer);

// 从USB鼠标PIPE中读数据
m_hInterruptTransfer = (*m_lpUsbFuncs->lpIssueInterruptTransfer)
(m_hInterruptPipe, MouseTransferCompleteStub, this,
USB_IN_TRANSFER | USB_SHORT_TRANSFER_OK, // 表示读数据
min(m_pInterface->lpEndpoints[0].Descriptor.wMaxPacketSize,
sizeof(m_pbDataBuffer)),
m_pbDataBuffer,
NULL);

if (m_hInterruptTransfer == NULL)
{
DEBUGMSG(ZONE_ERROR,(L"!USBMouse: Error in IssueInterruptTransfer\r\n"));
return FALSE;
}
else
{
DEBUGMSG(ZONE_TRANSFER,(L"USBMouse::SubmitInterrupt,Transfer:0x%X\r\n",
m_hInterruptTransfer));
}
return TRUE;
}

// 处理鼠标中断传输的数据
BOOL CMouse::HandleInterrupt()
{
DWORD dwError;
DWORD dwBytes;

DWORD dwFlags = 0;
INT dx = (signed char)m_pbDataBuffer[1];
INT dy = (signed char)m_pbDataBuffer[2];

BOOL fButton1= m_pbDataBuffer[0] & 0x01 ? TRUE : FALSE;
BOOL fButton2 = m_pbDataBuffer[0] & 0x02 ? TRUE : FALSE;
BOOL fButton3 = m_pbDataBuffer[0] & 0x04 ? TRUE : FALSE;
```

```
    if (!(*m_lpUsbFuncs->lpGetTransferStatus)(m_hInterruptTransfer,
&dwBytes,&dwError))
    {
    DEBUGMSG(ZONE_ERROR,(TEXT("!USBMouse: Error in
GetTransferStatus(0x%X)\r\n"),
    m_hInterruptTransfer));
    return FALSE;
    }
    else
    {
    DEBUGMSG(ZONE_TRANSFER,(TEXT("USBMouse::HandleInterrupt, hTransfer 0x%X
complete (%u bytes, Error:%X)\r\n"),
    m_hInterruptTransfer,dwBytes,dwError));
    }

    if (!SubmitInterrupt())
    return FALSE;

    if(dwError != USB_NO_ERROR)
    {
    DEBUGMSG(ZONE_ERROR,(TEXT("!USBMouse: Error 0x%X in interrupt
transfer\r\n"),dwError));
    return TRUE;
    }

    if(dwBytes < 3)
    {
    DEBUGMSG(ZONE_ERROR,(TEXT("!USBMouse: Invalid byte cnt %u from interrupt
transfer\r\n"),dwBytes));
    return TRUE;
    }

    if(dx || dy)
    dwFlags |= MOUSEEVENTF_MOVE;

    if(fButton1 != m_fPrevButton1)
    {
    if(fButton1)
    dwFlags |= MOUSEEVENTF_LEFTDOWN;
    else
    dwFlags |= MOUSEEVENTF_LEFTUP;
    }

    if(fButton2 != m_fPrevButton2)
    {
    if(fButton2)
    dwFlags |= MOUSEEVENTF_RIGHTDOWN;
    else
    dwFlags |= MOUSEEVENTF_RIGHTUP;
    }
```

```
if(fButton3 != m_fPrevButton3)
{
if(fButton3)
dwFlags |= MOUSEEVENTF_MIDDLEDOWN;
else
dwFlags |= MOUSEEVENTF_MIDDLEUP;
}

m_fPrevButton1 = fButton1;
m_fPrevButton2 = fButton2;
m_fPrevButton3 = fButton3;

DEBUGMSG(ZONE_EVENTS,
(TEXT("USBMouse event: dx:%d, dy:%d, dwFlags:0x%X (B1:%u, B2:%u, B3:%u)\r\n"),
dx,dy,dwFlags,fButton1,fButton2,fButton3));

// 通知系统产生鼠标事件
if (m_fReadyForMouseEvents)
mouse_event(dwFlags, dx, dy, 0, 0);
else
m_fReadyForMouseEvents = IsAPIReady(SH_WMGR);

return TRUE;
}


DWORD CALLBACK CMouse::MouseTransferCompleteStub(LPVOID lpvNotifyParameter)
{
CMouse * pMouse = (CMouse *)lpvNotifyParameter;
return(pMouse->MouseTransferComplete());
}

// 数据传输完毕回调函数
DWORD CMouse::MouseTransferComplete()
{
if (m_hEvent)
SetEvent(m_hEvent);
return 0;
}


ULONG CALLBACK CMouse::MouseThreadStub(PVOID context)
{
CMouse * pMouse = (CMouse *)context;
return(pMouse->MouseThread());
}

// USB鼠标线程
DWORD CMouse::MouseThread()
{
DEBUGMSG(ZONE_INIT,(TEXT("USBMouse: Worker thread started\r\n")));
SetThreadPriority(GetCurrentThread(), THREAD_PRIORITY_HIGHEST);
```

```
if (SubmitInterrupt())
{
while (!m_fClosing)
{
WaitForSingleObject(m_hEvent, INFINITE);

if (m_fClosing)
break;

if ((*m_lpUsbFuncs->lpIsTransferComplete)(m_hInterruptTransfer))
{
if (!HandleInterrupt())
break;
}
else
{
RETAILMSG(1,(TEXT("!USBMouse: Event signalled, but transfer not
complete\r\n")));
// The only time this should happen is if we get an error on the transfer
ASSERT(m_fClosing || (m_hInterruptTransfer == NULL));
break;
}
}
}
RETAILMSG(1,(TEXT("USBMouse: Worker thread exiting\r\n")));
return(0);
}
```

其实 USB 的驱动程序编写就是如此，类似的其他设备，例如打印机设备，就有 Bulk OUT PIPE，需要 Bulk 传输，那就需要了解 IssueBulkTransfer()的应用。当然如果是开发 USB Mass Storage Disk 的驱动，那就需要了解更多的协议，例如 Bulk-Only Transport 协议等。

6.2 USB 摄像头驱动介绍

6.2.1 USB Video Class 基础概念

USB 协议中，除了通用的软硬件电气接口规范等，还包含了各种各样的 Class 协议，用来为不同的功能定义各自的标准接口和具体的总线上的数据交互格式和内容。这些 Class 协议的数量非常多，最常见的比如支持 U 盘功能的 Mass Storage Class，以及通用的数据交换协议：CDC class。此外还包括 Audio Class、Print Class 等等。

理论上说，即使没有这些 Class，通过专用驱动也能够实现各种各样的应用功能。但是，正如 Mass StorageClass 的使用，使得各个厂商生产的 U 盘都能通过操作系统自带的统一的驱动程序来使用，这对 U 盘的普及使用起了极大的推动作用，制定其他这些 Class 也是为了同样的目的。

Video Class 协议的目的是给 USB 接口的视频设备提供一个统一的数据交换规范。最初版本是在 2003 年 9 月才添加到 USB Class 规范中的，1.1 的版本更是在 2005 年才发布。相比之下，Mass Storage Class 早在 1998 年就发布了。支持 Video Class 协议的多媒体芯片也是在 2005 年才陆续发布的。所以 USB 视频设备目前的现状是，在设备一端，多数依旧还采用原先的各种包含通用 USB 功能的多媒体处理芯片，主机端需要安装专用的驱动程序，基本上各个产品之间不具备兼容性。甚至对于操作系统而言，也只有在 XP 的 SP2 以后，才包含了对通用的 Video class 协议的支持。所以即使某些多媒体设备（比如 Logitech 最新的几款摄像头）包含了对 Video Class 的支持，在 Win 2000 等操作系统上依然需要安装驱动程序。不过，应该说使用 Video Class 无疑会是一个趋势，在相应的多媒体芯片陆续投入市场后，支持 Video Class 的多媒体设备应该会在一两年内迅速普及开来。

除了在硬件上通过相应的多媒体芯片支持 Video Class 的设备以外，对于包含了操作系统的智能手机，当然也可以在手机端通过驱动程序来实现对 Video Class 的支持，就好像原先支持任何一种专用的 USB 驱动一样。只不过数据交换的格式不是自己随意制订的，而是按照 Video Class 的规范来实现的。

6.2.2 USB Video Class 协议结构

（1） 设备拓扑结构

在拓扑结构上 Video Class 将视频设备抽象为几个主要的硬件功能模块：

- 输入端点 Input Terminal
- 输出端点 Output Terminal
- camera 端点 Camera Terminal
- 选择单元 Selector Unit
- 处理单元 Processing Unit
- 拓展单元 Extension Unit

图 6-1 是一幅 USB Video Camera Topology 的拓扑结构示例图。

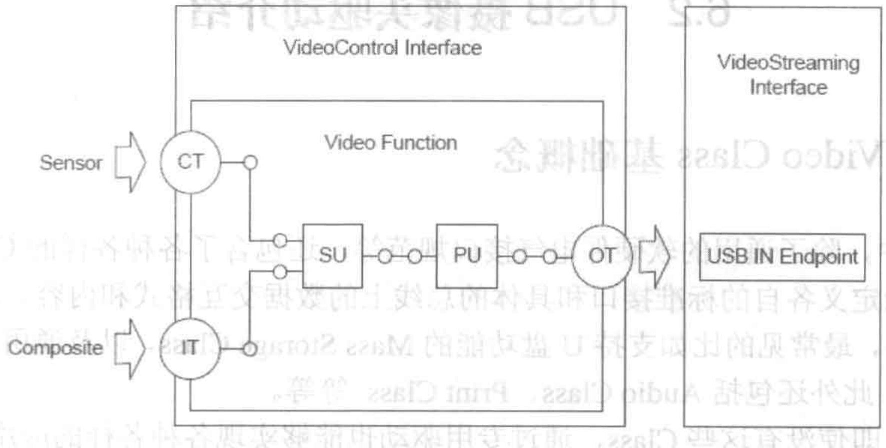


图 6-1

从 sensor 和另一个复合视频设备得到的数据流由 IT 和 CT 输入，经 SU 选择送 PU 处理，

再由OT绑定到指定的USB端点。最后由USB端点与主机交互将数据发送到host端。在实际设备中，可能没有其中的某些功能模块，也可能其中的几个模块都是由同一硬件来完成的。

（2）协议层次结构

图6-1中，左半部的框架组成了Video Class中的控制接口界面，右半部的框架组成了视频流传输接口界面。这两部分构成了Video Class的主要协议框架。

与Class相关的信息，当然是主机端通过向设备端获取描述符（Descriptor）来得到的，图6-2给出了一个Video Class协议描述符应用示例的布局图（Layout）。

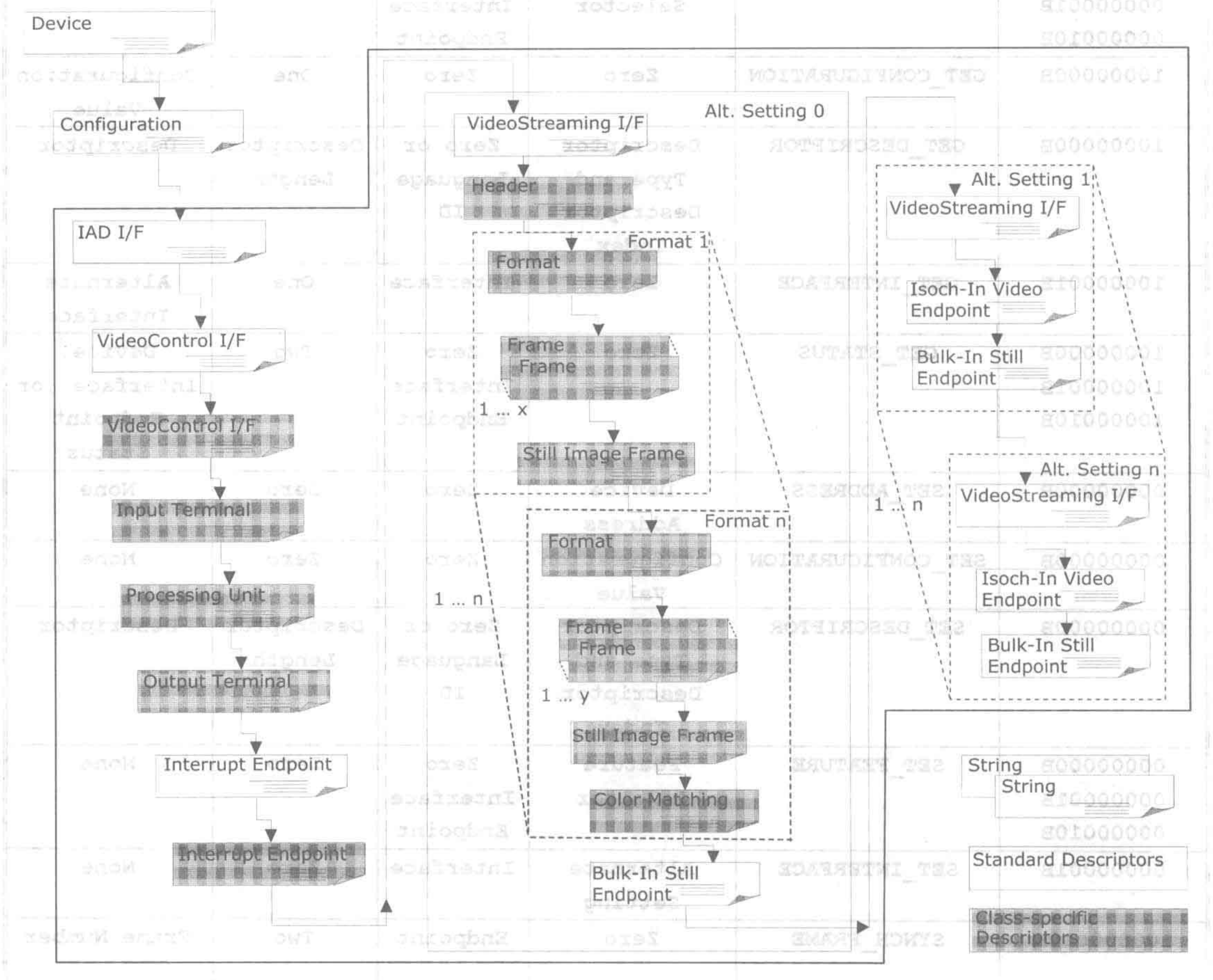


图6-2

可以看到，在描述符布局（Descriptor Layout）中，在标准描述符里，除了Device Descriptor、Configuration Descriptor、Interface Descriptor、Endpoint Descriptor、String Descriptor以外，还有一个USB 2.0协议中后期才新加的IAD，即Interface Association Descriptor，它用来描述多个相关Interface之间的关系。在Video Class中，IAD用来描述VideoControl Interface和VideoStreaming Interface之间的关系。

图中深色的部分就是Video Class协议相关的专用描述符（Class Specific Descriptor）。它主要用于对硬件图像采集和处理模块的物理拓扑结构和功能的描述，以及对视频传输格式（包括编码格式、码率等视频图像相关参数）的描述。

通过从设备处获得这些描述符，主机可以得知视频设备端的结构及其所支持的功能。而控制这些功能模块，对数据源和数据流进行配置，则需要通过 Request 来完成。

Request 是由主机向设备端发起的功能请求，包括所有 USB 设备都需要支持的 Standard Device Requests 和与 Class 相关的 Class Specific Requests。

图 6-3 列出了 USB Spec 中规定的标准 Request。

bmRequestType	bRequest	wValue	Windex	wLength	Data
00000000B 00000001B 00000010B	CLEAR_FEATURE	Feature Selector	Zero Interface Endpoint	Zero	None
10000000B	GET_CONFIGURATION	Zero	Zero	One	Configuration Value
10000000B	GET_DESCRIPTOR	Descriptor Type and Descriptor Index	Zero or Language ID	Descriptor Length	Descriptor
10000001B	GET_INTERFACE	Zero	Interface	One	Alternate Interface
10000000B 10000001B 10000010B	GET_STATUS	Zero	Zero Interface Endpoint	Two	Device, Interface ,or Endpoint Status
00000000B	SET_ADDRESS	Device Address	Zero	Zero	None
00000000B	SET_CONFIGURATION	Configuration Value	Zero	Zero	None
00000000B	SET_DESCRIPTOR	Descriptor Type and Descriptor Index	Zero or Language ID	Descriptor Length	Descriptor
00000000B 00000001B 00000010B	SET_FEATURE	Feature Selector	Zero Interface Endpoint	Zero	None
00000001B	SET_INTERFACE	Alternate Setting	Interface	Zero	None
100000010B	SYNCH_FRAME	Zero	Endpoint	Two	Frame Number

图 6-3

这其中，有一部分 Request 是由 USB 控制芯片在硬件一级就直接完成的应答工作，比如 SET_ADDRESS，有些则需要由软件来做进一步的处理，比如 Set_Configuration。软硬件的这种任务的划分还与具体的硬件芯片相关。因为这部分是标准协议相关的，本文就不详述。

Class Specific Requests 的数据结构 Layout 与标准 Request 是一样的，只是内容不同而已。VideoClass 的 Class Specific Requests 主要根据 Interface 分为两类，其下又根据具体功能模块做进一步的划分：

VideoControl Requests

Camera Terminal Control Requests

Selector Unit Control Requests
Processing Unit Control Requests
Extension Unit Control Requests
VideoStreaming Requests
Interface Control Requests

这其中，Interface Control Requests 因为是用来在主机和设备之间协商数据交互格式和分辨率、流量等信息的，所以一般来说是必须实现的。

而 Camera Terminal Control Requests 和 Processing Unit Control Requests 中的内容，则是目前常用的即时通信软件如 MSN / QQ 等在其视频控制界面上集成的控制参数。

其中，Camera Terminal Control Requests 包含了对曝光时间、曝光模式、对焦、变焦、平移等传感器（sensor）获取数据阶段时的参数控制。而 Processing Unit Control Requests 中则包含了亮度、增益、色调、对比度、白平衡等 sensor 在获取到图像数据后，图像处理阶段的相关参数。

不过，以上两者的划分在硬件层次并不是绝对的。很多参数的控制在 sensor 硬件级别上是同一层次的。不过，将这些功能抽象成这两类，正如在硬件的拓扑结构上将功能模块抽象出来一样，有利于通用化的程序设计。

6.3　微软的 USB Video Class 驱动使用

只要支持 USB Video Class 协议的设备，就可以像支持 USB Mass Storage 协议的设备一样，实现通用驱动，也就是 Windows XP 上所说的“无驱型”摄像头。

微软在 2007 年 6 月为 WINCE6 发布了一款支持 USB Video Clas1.1 协议的摄像头驱动程序。

下载地址：

http://www.microsoft.com/downloads/details.aspx?FamilyID=2ef087c0-a4ae-42cc-abd0-c466787c11f2&DisplayLang=en

安装过程如图 6-4、图 6-5、图 6-6、图 6-7 所示，按照向导默认安装即可，它并不是一个独立的软件，而是作为 WINCE6 的一个组件存在的。

图 6-4

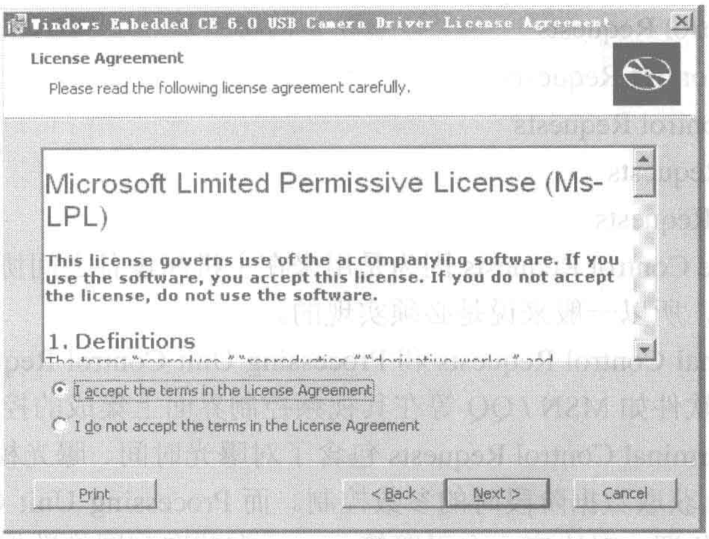


图 6-5

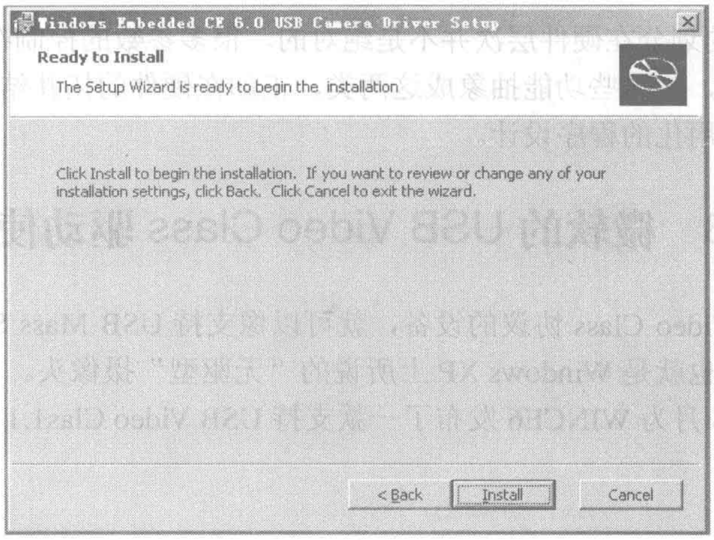


图 6-6

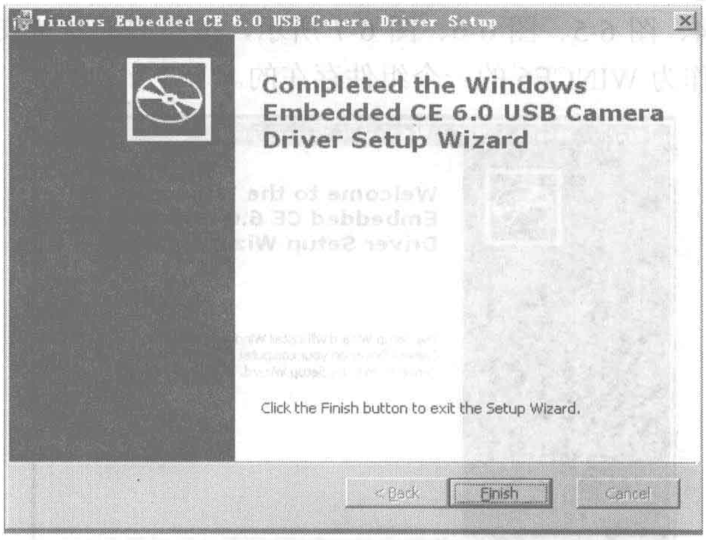


图 6-7

安装完成之后，该组件出现在 PLATFORM BUILDER 平台上的位置如图 6-8 所示。

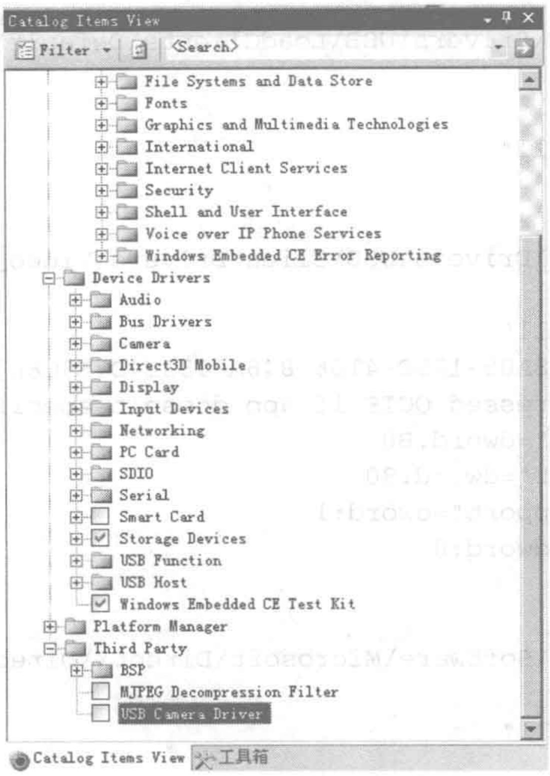


图 6-8

源代码所在文件夹位置：“C:\WINCE600\PUBLIC\ThirdParty\Catalog\USBCam”。

注册表设置如下：

```
;-------------------------------------------------------------------------
; <copyright file="USBCam.reg" company="Microsoft">
;  Copyright (c) Microsoft Corporation.  All rights reserved.
;
;    The use and distribution terms for this software are covered by the
;    Microsoft Limited Permissive License (Ms-LPL)
;    which can be found in the file MS-LPL.txt at the root of this distribution.
;    By using this software in any fashion, you are agreeing to be bound by
;    the terms of this license.
;
;    THE SOFTWARE IS LICENSED "AS-IS" WITH NO WARRANTIES OR INDEMNITIES.
;
;    You must not remove this notice, or any other, from this software.
; </copyright>
;
; <summary>
;USB camera driver for Windows Embedded CE 6.0
; </summary>
;-------------------------------------------------------------------------
;=======================================================================
; USB camera driver for Windows Embedded CE 6.0
;=======================================================================
```

```
[HKEY_LOCAL_MACHINE\Drivers\USB\LoadClients\Default\Default\Default\Video_
Class]
    "Prefix"="CAM"
    "Dll"="usbcam.dll"

[HKEY_LOCAL_MACHINE\Drivers\USB\ClientDrivers\Video_Class]
    "Prefix"="CAM"
    "Dll"="usbcam.dll"
    "IClass"="{CB998A05-122C-4166-846A-933E4D7E3C86}"
; Default to Uncompressed QCIF if app doesn't specify
    "PreferredWidth"=dword:B0
    "PreferredHeight"=dword:90
    "UncompressedSupport"=dword:1
    "MJPEGSupport"=dword:0

[HKEY_LOCAL_MACHINE\Software\Microsoft\DirectX\DirectShow\Capture]
    "Prefix"="PIN"
    "Dll"="usbcam.dll"
    "Order"=dword:1
    "Index"=dword:1
    "IClass"="{C9D092D6-827A-45E2-8144-DE1982BFC3A8}"

[$(PCI_BUS_ROOT)\Template\ehci]
   "PhysicalPageSize"=dword:200000 ; increase the physical buffer for
isochronous transfers
```

MSDN 上的参考文档地址：http://msdn.microsoft.com/en-us/library/aa924811.aspx。

该驱动程序的详细说明如下所示（以下摘自驱动程序的“release notes”，为确保其原意故不作翻译）。

```
;---------------------------------------------------------------------------
; <copyright file="ReleaseNotes.txt" company="Microsoft">
;  Copyright (c) Microsoft Corporation.  All rights reserved.
;
;    The use and distribution terms for this software are covered by the
;    Microsoft Limited Permissive License (Ms-LPL)
;    which can be found in the file MS-LPL.txt at the root of this distribution.
;    By using this software in any fashion, you are agreeing to be bound by
;    the terms of this license.
;
;    THE SOFTWARE IS LICENSED "AS-IS" WITH NO WARRANTIES OR INDEMNITIES.
;
;    You must not remove this notice, or any other, from this software.
; </copyright>
;
; <summary>
;    USB camera driver for Windows Embedded CE 6.0
; </summary>
;---------------------------------------------------------------------------
;=========================================================================
; USB camera driver for Windows Embedded CE 6.0
```

```
;===========================================================================

Version 1.0
April 26th 2007

-------------------------------------------------------------------------------
---
OVERVIEW
-------------------------------------------------------------------------------
---

This device driver adds support for USB cameras to Windows Embedded CE 6.0.
It enables still and video capture from cameras which comply with the USB
Video Class specification (version 1.1).

The driver is distributed as shared source under the Microsoft Limited
Permissive License, and must be compiled for your target platform.


!! THIS DRIVER IS CONSIDERED A 揚OWER TOY?AND MAY NOT BE SUITABLE FOR
!! PRODUCTION USE WITHOUT FURTHER TESTING OR MODIFICATION.


This driver implements the camera interface introduced in Windows Mobile 5.0,
and now available in CE 6.0.  In other words, the driver is meant to be
used as the source filter in a DirectShow capture graph, rather than as a
stand-alone stream-based driver.

For more information on using this driver with DirectShow, see the topic
"Video Capture" in the CE documentation:

    http://msdn2.microsoft.com/en-us/library/aa924811.aspx


The driver supports both USB 1.x and 2.0 (high-speed) cameras.  The driver will
expose both uncompressed and MJPEG modes, if supported by the camera.

MJPEG mode generally requires pairing the driver with a DirectShow transform
filter (jpeg2rgb.dll) which is also included in this download.


Not Implemented:
----------------
- USB audio


Hardware Dependency:
--------------------
- DMA-capable USB host controller recommended
- USB 2.0 recommended for uncompressed mode
- Fast CPU recommended for MJPEG mode
```

```
- Tested on the following:

    Platforms (x86)
    ---------------
    CEPC:             1.8GHz Duron, USB 2.0
    iCOP eBox-2300:   200MHz Vortex86 SoC, USB 1.1

    Cameras
    -------
    Microsoft LifeCam NX-6000
    Logitech QuickCam Pro 5000

    NOTE:
    -----
    The driver has only been tested on x86, but has no known platform
dependencies.


Software Dependency:
--------------------
- Windows Embedded CE 6.0
    - USB support
    - DirectShow


SYSGEN Dependencies:
--------------------
Needs following SYSGENS set, in addition to any required by your application:
- SYSGEN_USB
- SYSGEN_DSHOW_CAPTURE
- SYSGEN_DSHOW_DMO
- SYSGEN_DSHOW_VIDREND

The following are only required for jpeg2rgb.dll:
- SYSGEN_DCOM_STG
- SYSGEN_IMAGING=1
- SYSGEN_IMAGING_BMP_DECODE=1
- SYSGEN_IMAGING_BMP_ENCCODE=1
- SYSGEN_IMAGING_JPG_ENCCODE=1
- SYSGEN_IMAGING_JPG_DECODE=1

The following is only required for CameraDshowApp.exe (ships with the
Windows Embedded CE 6.0 shared source; not included in this download):
- SYSGEN_DCOM


-------------------------------------------------------------------------------
---
PACKAGE CONTENTS
-------------------------------------------------------------------------------
---
```

```
   usbcam        - source for the USB Video Class driver
   mjpegfilter   - source for the MJPEG -> RGB24 DShow Transform Filter

   --------------------------------------------------------------------------
---
   BUILD INSTRUCTIONS
   --------------------------------------------------------------------------
---

   Building the Driver from IDE (Recommended):
   ------------------------------------------------

   - Verify the source is present in
%_WINCEROOT%\Public\ThirdParty\Catalog\USBcam

   - Open your OSDesign in PB (if it was already open, refresh the catalog in PB)

   - The Usbcam and Mjpegfilter will show up as catalog items under the Third Party
     folder. Select these catalog items.

     NOTE: Selecting the catalog items will cause PB to automatically add usbcam
           and mjpegfilter as subprojects to your OSDesign

   - Rebuild your project from the IDE (Build Menu -> Rebuild Solution)

     NOTE: The catalog items automatically pull in the required SYSGEN settings
as
           specified in the "SYSGEN Dependencies" section of this document. Hence
           a Rebuild of the solution is recommended so that the required components
           get sysgen'd correctly.

   - The built image will now contain the usbcam driver and the mjpeg filter, along
     with the required components and registry keys.


   Building the Driver from Command Line (Optional):
   ------------------------------------------------

      NOTE: The following assumes you are building from a command-line build
      window for a CE 6.0 CEPC BSP. If your platform is different, substitute
      your platform name in place of CEPC


   - Unzip the archive under %_WINCEROOT%\platform\CEPC\src\drivers\
   - Make sure the new folders are not marked read-only
   - Add usbcam and mjpegfilter to %_WINCEROOT%\platform\CEPC\src\drivers\dirs

   - Add the following to the end of platform.reg :
      ; @CESYSGEN IF DIRECTX_MODULES_QUARTZ
      ; @CESYSGEN IF CE_MODULES_USBD
        ; Camera driver
```

```
        IF BSP_NOCAMERA !
            #include "$(_TARGETPLATROOT)\src\drivers\usbcam\usbcam.reg"
            #include
"$(_TARGETPLATROOT)\src\drivers\mjpegfilter\jpeg2rgb.reg"
        ENDIF BSP_NOCAMERA !
    ; @CESYSGEN ENDIF CE_MODULES_USBD
    ; @CESYSGEN ENDIF DIRECTX_MODULES_QUARTZ

  - Add the following to the end of platform.bib:

    ; @CESYSGEN IF DIRECTX_MODULES_QUARTZ
    ; @CESYSGEN IF CE_MODULES_USBD
        IF BSP_NOCAMERA !
            usbcam.dll      $(_FLATRELEASEDIR)\usbcam.dll               NK SHK
            jpeg2rgb.dll    $(_FLATRELEASEDIR)\jpeg2rgb.dll             NK SH
        ENDIF BSP_NOCAMERA !
    ; @CESYSGEN ENDIF CE_MODULES_USBD
    ; @CESYSGEN ENDIF DIRECTX_MODULES_QUARTZ

    NOTE:
        usbcam.dll is a kernel-mode driver
        jpeg2rgb.dll is a user-mode component
  - Set the required SYSGEN variables
  - Build the driver, filter, and image.
    E.G. from a build window type "blddemo clean cleanplat -q"
  - Verify that usbcam.dll and jpeg2rgb.dll are in the image.
    E.G. from a build window type "viewbin -toc %_FLATRELEASEDIR%\nk.bin"
  - Verify the information from usbcam.reg and jpeg2rgb.reg is in the
  image (look in %_FLATRELEASEDIR%\reginit.ini)

  When you attach a camera, usbcam.dll will load and query the camera.
  If the camera complies with the USB Video Class specification, usbcam.dll
  will initialize the camera and query its supported properties and image
  formats.

  If the driver does not properly detect your camera, you may need to modify
  USBDeviceAttach() and ParseStreamInterfaces() in usbcode.cpp.


-------------------------------------------------------------------------------
---
  USING / TESTING THE DRIVER
-------------------------------------------------------------------------------
---

  Make sure you have installed the Windows Embedded CE 6.0 Shared Source.
  An ad-hoc camera test application is available, which uses DShow (and the
  optional MJPEG filter) to display video.  Source is available at:

    Wince600\private\test\multimedia\directx\dshow\camera\cameraapp

  To run CameraDshowApp:
```

```
   1) Image should include SYSGEN_SHELL (enables the Platform Builder Target
      Control Window)
   2) Copy CameraDshowApp.exe to your flat release dir.
   3) Plug in the Camera.  Driver should load and query camera properties/formats.
   4) From the Target Control window, type "s CameraDshowApp"
   5) In the dialog that appears, select CAM1.  If it's missing, driver failed
to
      recognize your device (or isn't in the image).
   6) Uncheck everything except "VideoCapture" and "VideoRenderer".
   7) Click OK to start video streaming.

   Also refer to the CETK:

   Camera and DirectShow Integration Test:
       http://msdn2.microsoft.com/en-us/library/aa935155.aspx

   Camera Driver Data Structure Verification Test:
       http://msdn2.microsoft.com/en-us/library/aa934069.aspx

   Camera Driver I/O Test:
       http://msdn2.microsoft.com/en-us/library/aa934054.aspx

   Camera Performance Test:
       http://msdn2.microsoft.com/en-us/library/aa934370.aspx

   ---------------------------------------------------------------------------
---
   KNOWN ISSUES / NOTES
   ---------------------------------------------------------------------------
---

   - The driver resolution is configurable via the registry.  At present,
     changing the resolution via DShow is unreliable.

   - The driver may be configured to support MJPEG, uncompressed, or both.

     In usbcam.reg, modify one or both of the following lines:

       [HKEY_LOCAL_MACHINE\Drivers\USB\ClientDrivers\Video_Class]
           "UncompressedSupport"=dword:1
           "MJPEGSupport"=dword:0

     Uncompressed video requires USB 2.0 to be useful (bandwidth-intensive).
     MJPEG video requires a very fast CPU to be useful (decode-intensive).
     In both cases, a DMA-capable USB controller is strongly recommended.

     While the driver can deliver MJPEG at full frame rates, frames
     may be dropped if the CPU (MJPEG filter) can't keep up.  The current filter
     code needs optimization.

     Typical frame rates (measured on a 1.8GHz CEPC w/ USB 2.0 and RageXL graphics):
       Uncompressed video, 640x480:    30 fps
```

```
    MJPEG video,      640x480:   12 fps (18fps dropped due to slow MJPEG
filter)

  - If taking a still capture while streaming video, still capture will fail
    unless the requested resolution is the same as the streaming video resolution.

  - In MJPEG mode, the driver prefixes a JPEG header and Huffman segment to each
    frame read from the camera.  These frames comform to the JPEG File Interchange
    Format; they may be read by any JPEG-aware application or library.
    The MJPEG DirectShow filter is one example.

    In fact, you should be able to save the frame with a .jpg extension and open
    it in an application such as Internet Explorer.


  - Usbcam.dll will be loaded for every USB device attached to your system.
    It queries the device and if it finds a USB Video Camera, takes ownership
of it.
    Otherwise, the driver immediately unloads.

    To restrict the driver to a more narrow range of devices, modify usbcam.reg:

[HKEY_LOCAL_MACHINE\Drivers\USB\LoadClients\Default\Default\Default\Video_Clas
s]

      (replace Default\Default\Default with the correct vendor and device IDs
for
       your camera)


  - There are many opportunities for performance optimization in the MJPEG filter.
    E.G. reducing data copying.  See CJPEGFilter::Transform()


  ----------------------------------------------------------------------------
---
  ADDITIONAL RESOURCES
  ----------------------------------------------------------------------------
---

  In the future, we may post articles and sample code for this project at the
  Windows CE Base Team Blog:

     http://blogs.msdn.com/ce_base/
```

第7章 显卡驱动

显卡驱动的开发是极其困难的，尽管 WINCE 的显卡驱动编程较之 Windows 操作系统已有很大简化，但其难度依然巨大。并且有许多显卡硬件规格并没有详细公布，想要将显卡的硬件性能充分发挥是极其困难的。本章将简要介绍 GDI 驱动开发方式，但不建议自己开发。对于 Intel 的一系列集成显卡，Intel 已经提供了驱动程序，直接调用即可，并且可以发挥强大的显卡性能。

7.1 WINCE 显卡驱动编程介绍

在最早的 WINCE 版本中，图形设备接口（GDI）直接与显示设备的硬件相连接。在新版本中，GDI 与一个或多个显示驱动程序连接，这些显示驱动程序依次与显示设备的硬件相连接。这种显示结构使 Windows CE 能够使用各种显示设备，而不需要每个设备都有一个根据硬件编程的接口程序。

首先，我们将考虑一下 Windows CE 显示设备驱动程序成功的几个因素。这里我们先讨论一下显示设备驱动程序应该实现的接口功能，和可以用于简化显示设备驱动程序开发任务的图形基本引擎类。同时也讲一下设备驱动程序对 GDI 的支持。其次，我们将给出一个详细列表，列出 Windows CE 所支持的每种像素灰度所对应的显示缓冲格式，并讲述一下显示设备的硬件。由于显示设备驱动程序是 GDI 和显示设备之间的纽带，硬件有效地支持了显示驱动程序，也就有效地支持了 GDI。

像 Windows CE 的其他部分一样，显示驱动程序接口（DDI）是 Windows NT DDI 的一个子集。Windows CE 仅仅使用了来自 Windows NT DDI 的基本图形引擎函数和驱动程序函数。Windows CE 和 Windows NT 之间的不同，造成了 Windows CE 显示设备驱动程序的下列差别：

Windows CE 显示设备驱动程序总是提供一组相同的功能，GDI 不需要查询关于驱动程序能力的任何信息。

Windows CE 显示设备驱动程序不能拒绝十分复杂的操作，它反过来调用 GDI 将复杂的操作分解成简单的步骤。因为 Windows CE 所有的显示设备驱动程序支持相同的功能，在调用显示设备驱动程序之前，GDI 第一步可以将复杂的操作分解为简单的操作。

Windows CE 显示设备驱动程序被编译为动态连接库(.dll 文件)而不是编译成库文件(.lib 文件)。

Windows CE 所有的显示设备驱动程序必须实现一组 DDI 函数，GDI 调用这组函数初始化显示设备驱动程序和将图形输出到显示设备上。除了 DDI 函数，有一组被称为基本图形引擎（GPE）类的 Microsoft Visual C++®类，显示设备驱动程序可以使用它们方便地进行硬件加速。使用 GPE 类及其方法实现对基于 S3Trio64 显示设备硬件的加速，便是实现显示设备驱动程序加速的一个例子。如果你显示设备的硬件使用不同的芯片组，可以改变 GPE 方法的实现方式以适应自己的硬件特性。

注意，是否使用 GPE 类是随意的。在编写显示设备驱动程序时，可以不使用它们，只不过代价是你的 DDI 功能的实现将更加复杂。注意微软提供的 GPE 类要求你所用显示设备的硬件必须有一个连续的帧缓冲区。如果你显示设备的硬件不是如此（例如，如果它使用一个固定大小的移动窗口来存取整个显示内存），那么它就不能使用 GPE 类。要得到更多的信息，请参阅本文的“Windows CE 显示设备硬件推荐”部分，尤其是“脏矩形区的驱动”子专题。

Windows CE 显示设备的驱动在许多方面与一般的设备驱动不同。最主要的不同在于它们没有明显的输入/输出流接口，因此，它们不能被设备管理器管理，它们也不能调用 RegisterDevice。结果，没有特别的设备文件或其他文件系统途径与显示设备驱动程序相对应。当应用程序需要使用显示设备驱动程序，而使用显示设备的.dll 文件名调用 CreateDC 时，显示设备驱动程序被装入。这时，Windows CE 装入显示设备驱动程序并进行相应的初始化，以便于将该设备的上下文返回给发出调用的应用程序。当然，默认的显示设备驱动程序被自动装入。

表 7-1 中列出了显示设备和打印设备驱动程序的 DDI 函数。显示设备驱动程序应当实现这里列出的所有显示设备 DDI 函数；打印设备驱动程序应当实现所有列出的打印 DDI 函数。但是，只有 DrvEnableDriver 必须在显示设备驱动程序的 DLL 中表示出。因此，只有 DrvEnableDriver 必须遵循这个名字；而其他函数可以根据用户的需要进行调用，因为它们是通过 DrvEnableDriver 返回的函数指针表示出来的。无论它们被谁调用，它们都必须遵照在 WinDDI.H 中定义的原型。

表 7-1

函 数	目 的
DrvAnyBlt	以弹性或透明的方式进行位块传送
DrvBitBlt	以剪贴或屏蔽的方式进行通常的位块传送
DrvContrastControl	使用软件调节显示设备的硬件设置
DrvCopyBits	将创建的 GDI 打印区发送给打印设备驱动程序
DrvCreateDeviceBitmap	创建并管理位图
DrvDeleteDeviceBitmap	删除设备位图
DrvDisableDriver	提醒驱动程序 GDI 不再需要它并准备卸载它
DrvDisablePDEV	提醒驱动程序 GDI 不再需要特定的打印或显示设备
DrvDisableSurface	提醒驱动程序 GDI 不再需要特定的绘图平面
DrvEnableDriver	由驱动程序表示初始化入口点，并将其他 DDI 函数指针返回给 GDI
DrvEnablePDEV	将一个 PDEV（物理显示设备的逻辑表示）返回给 GDI
DrvEnableSurface	创建一个绘图平面并将它与一个 PDEV 关联
DrvEndDoc	将完成打印所需的控制信息发送给一个文档
DrvFillPath	用画笔填充一个路径
DrvGetMasks	获取显示设备当前模式的颜色掩码
DrvGetModes	列出显示设备所支持的显示模式
DrvMovePointer	确保不受 GDI 干扰的移动指针
DrvPaint	用画笔绘制特定的区域
DrvPowerHandler	被调用来处理能量高和能量低的提示
DrvQueryFont	获取字体公制单位信息
DrvRealizeBrush	用 GDI 指定的参数创建一个画笔
DrvRealizeColor	将一个 RGB 颜色映射到最接近的设备所支持的一个有效的颜色上
DrvSetPalette	设置显示设备调色板
DrvSetPointerShape	设置指向一个新的形状的指针并更新显示

（续表）

函　数	目　的
DrvStartDoc	将启动打印所需的控制信息发送给一个文档
DrvStartPage	将启动打印所需的控制信息发送给一个新的页面
DrvStrokePath	绘制一个路径
DrvTransparentBlt	以透明方式进行位块传送
DrvUnrealizeColor	以该显示设备的格式将一个颜色映射给一个 RGB 值

示例显示设备驱动程序使用 GPE 类。尽管 GPE 类是任选的，但是使用它可以极大地方便对显示设备驱动程序的编写。如果用户使用 GPE 类，那么用户只需提供使自己的显示设备硬件功能正常和实现加速所需的新代码。GPE 类要求自己的显示设备的硬件使用一个连续的帧缓冲区。这就是说，显示设备内存必须位于一个邻近的内存范围内。修改 GPE 类来使用一段不相邻的帧缓冲区则需要用户付出很大的努力。

要创建一个基于 GPE 类的显示设备驱动程序，需按照下列步骤：

- 为你的工程创建一个目录。
- 从一个示例驱动程序目录中将文件复制到你的工程目录中。
- 在这些文件中对特定的设备名进行全局替换。
- 改变 Config.CPP 以便于将你的显示设备设置为一种线性帧缓冲区模式。
- 禁用所有特定硬件的加速。
- 编译并测试这个没有加速的驱动程序。GPE 将使用软件仿真生成输出。
- 添加你自己的硬件加速代码。

Windows CE GDI 提供了一些服务来支持显示设备，以预先定义的结构的方式并带有作用在其上的函数，和一些独立的 C 函数，如表 7-2 所示。预先定义的结构提供了对画笔、剪贴区域、调色板、笔画、填充方式和转换的支持。独立方式的 C 函数提供了对设备位图和平面的支持。

表 7-2

结构和函数	目　的
BRUSHOBJ	用于表示实心或模式笔画和填充操作的画笔结构
BRUSHOBJ_pvAllocRbrush	给画笔分配内存的函数
BRUSHOBJ_pvGetRbrush	重新获取指向指定笔画指针的函数
CLIPOBJ	表示剪贴区域的结构
CLIPOBJ_bEnum	列举在剪贴区域中的剪贴矩形的函数
CLIPOBJ_cEnumStart	给剪贴区域中列举出的矩形设置参数的函数
EngCreateDeviceBitmap	用于使 GDI 为设备位图创建句柄的函数
EngCreateDeviceSurface	用于使 GDI 创建将由显示设备驱动程序管理的设备平面的函数
EngDeleteSurface	用于通知 GDI 显示设备驱动程序不再需要该设备平面的函数
PALOBJ_cGetColors	将颜色复制到调色板的函数
PATHDATA	存储部分绘画轨迹的结构
PATHOBJ_bEnum	列举绘画轨迹中 PATHDATA 记录的函数
PATHOBJ_vEnumStart	使绘画轨迹准备好列举它的组件线段的函数
PATHOBJ_vGetBounds	返回绘画轨迹所限制的矩形的函数
XLATEOBJ	用于从一个调色板向另一个调色板传送颜色的结构
XLATEOBJ_cGetPalette	重新从一个索引指出的调色板获取颜色的函数

Windows CE GDI 支持带有多种颜色灰度和颜色模式的显示设备，从仅用一位表示的颜色到调色板调制出的颜色，直到真 32 位 RGB。每一种格式中又支持几种不同的像素排列方式，这取决于对显示内存的访问是否支持字节方式、双字节字方式或四字节 DWORD 方式。

所有显示缓冲区格式都假定显示设备上像素的顺序是从左到右和从上到下的。也就是说，像素（0，0）位于显示设备的左上角，像素（width-1，height-1）位于显示设备的右下角。

每个像素用一位表示的格式 One-Bit-Per-Pixel Format。

每个像素用一位表示的格式是针对简单的黑白显示设备的。0 表示黑，1 表示白。由多个像素组成字节，如像素（0，0）位于显示内存第一个字节的最高位上。这种格式的内存可以被排列成下列方式，如图 7-1 所示。

Byte Access

Byte0		Byte1		Byte2		Byte3	
B07	B00	B07	B00	B07	B00	B07	B00
P00	P07	P08	P0F	P10	P17	P18	P1F

Word Access

Word 0				Word 1			
111F	1118	1117	1110	011F	0118	0117	0110
P08	P0F	P00	P07	P18	P1F	P10	P17

DWord Access

DWord 0							
111F	1118	1117	1110	011F	0118	0117	0110
P18	P1F	P10	P17	P08	P0F	P00	P07

图 7-1

每个像素用两位表示的格式：尽管任意 4 条目的调色板都可以使用，但是每个像素用两位表示的格式是 4 级灰度显示设备的一种典型用法。各灰度级所表示的内容可以根据表 7-3 所示确定。

表 7-3

Bit 1	Bit 0	Gray Level
0	0	黑
0	1	深灰
1	0	浅灰
1	1	白

这种格式的内存可以被排列成如图 7-2 所示的方式。

Byte Access

Byte0		Byte1		Byte2		Byte3	
B07	B00	B07	B00	B07	B00	B07	B00
P00	P01	P02	P03	P04	P05	P06	P07

Word Access

Word 0				Word 1			
111F	1118	1117	1110	011F	0118	0117	0110
P04	P07	P00	P03	P0C	P0F	P08	P0B

DWord Access

DWord 0							
111F	1118	1117	1110	011F	0118	0117	0110
P0C	P0F	P08	P0B	P04	P07	P00	P03

图 7-2

每个像素用四位表示的格式：每个像素用四位表示的格式是常用的一种调色板格式。帧

缓冲区本身可以被实现为将两个像素组成一个字节，或每个字节表示一个像素两种方式。这种格式的内存可以被排列成如图 7-3 所示的方式。

Byte Access

Byte0		Byte1		Byte2		Byte3	
B07	B00	B07	B00	B07	B00	B07	B00
P00	P01	P02	P03	P04	P05	P06	P07

Word Access

Word 0				Word 1			
111F	1118	1117	1110	011F	0118	0117	0110
P03	P02	P00	P01	P06	P07	P04	P05

DWord Access

DWord 0							
111F	1118	1117	1110	011F	0118	0117	0110
P06	P07	P04	P05	P03	P03	P00	P01

图 7-3

如果读者选择使用每字节表示一个像素的方式，那么驱动程序应当将显示方式表示成每个像素使用 8 位的 16 色调色板方式。每个字节中相对应的位应当是位于低阶的半字节，而高阶的半字节应当总是零。

每个像素用八位表示的格式可以很理想地用一个软件改变的调色板来将 8 位的值映射到 24 位的颜色，如图 7-4 所示。由于执行、兼容性和图像质量的原因，微软推荐使用默认的 Windows CE 调色板。

Byte Access

Byte0		Byte1		Byte2		Byte3	
B07	B00	B07	B00	B07	B00	B07	B00
P00		P01		P02		P03	

Word Access

Word 0				Word 1			
111F	1118	1117	1110	011F	0118	0117	0110
P01		P00		P03		P02	

DWord Access

DWord 0							
111F	1118	1117	1110	011F	0118	0117	0110
P03		P02		P01		P00	

图 7-4

每个像素用 15 位或 16 位表示的格式是一种掩码的格式，并且不被调色。对于每个像素用 15 位或 16 位表示的格式，像素是用两字节的字存储的；每个像素用 15 位表示的格式浪费了每个字的最高位。微软推荐使用如表 7-4 所列掩码来提取红、绿、蓝数值。

表 7-4

颜　色	Fifteen-Bit （5-5-5 RGB）	Sixteen-Bit （5-6-5 RGB）
红	0x7C00	0xF800
绿	0x3E00	0x07E0
蓝	0x001F	0x001F

正如每个像素用 15 位表示的掩码所显示的，每个字的低 15 位包含该像素的数据。未用的位应当是零。这种格式的内存可以被排列成如图 7-5 所示的方式。

Word Access

Word 0				Word 1			
111F	1118	1117	1110	011F	0118	0117	0110
P00				P01			

DWord Access

DWord 0							
111F	1118	1117	1110	011F	0118	0117	0110
P01				P00			

图 7-5

每个像素用 24 位表示的格式是一种真彩色的格式，其中每个像素用红、绿、蓝各 8 位的方式存储。这种格式的优点是图像质量非常好，因为每个像素恰好占 3 个字节，它们可以被一起放置而不会浪费内存；但缺点是有半数的像素跨越 DWORD 边界，在存取和解码时会有执行效率上的损失。这种格式的内存可以被排列成如图 7-6 所示的方式。

Byte Access

Byte0		Byte1		Byte2		Byte3	
B07	B00	B07	B00	B07	B00	B07	B00
P00:blue channel		P00:green channel		P00:red channel		P01:blue channel	

图 7-6

每个像素用 32 位表示的格式是另一种真彩色的格式。这种格式不会使像素产生跨越 DWORD 边界的情况，但是在内存使用上效率低。在这种格式中有两种方式安排颜色信道。一种方式是将蓝色放在每个像素最不重要的字节中，另一种方式将红色放在最不重要的字节中。这些选项与 PAL_BGR 和 PAL_RGB 方式相对应。读者可以使用如图 7-5 所列掩码来从每个像素中提取红、绿、蓝和 alpha 信道。

表 7-5

Color	PAL_RGB Mask	PAL_BGR Mask
Red	0x000000FF	0x00FF0000
Green	0x0000FF00	0x0000FF00
Blue	0x00FF0000	0x000000FF

这种格式的内存可以被排列成如图 7-7 所示的方式。

DWord Access: PAL_RGB

DWord 0							
111F	1118	1117	1110	011F	0118	0117	0110
Unused		P00:blue channel		P00:green channel		P00:red channel	

DWord Access: PAL_BGR

DWord 0							
111F	1118	1117	1110	011F	0118	0117	0110
Unused		P00:red channel		P00:green channel		P00:blue channel	

图 7-7

对用于 Windows CE 操作系统的显示设备硬件，微软有一些推荐标准。这些推荐标准有益于提高执行效率和方便读者进行显示设备驱动程序的开发。即使读者的硬件与这些推荐的标准并不一致，仍然可以编写功能完善的显示设备驱动程序。

微软强烈推荐读者的显示设备硬件要使用线性帧缓冲区，读者应当能够读写缓冲区。读者全部的显示内存应当是连续的，更好的情况下，应当是一个线性存取的窗口，并能够覆盖整个帧缓冲区。对于不符合这个推荐标准的硬件，如果读者选择使用 GPE，那么将需要对

GPE 类做实质性的修改。

用户的显示设备硬件还应当使用一种被支持的像素格式、压缩和排列的组合。显示设备硬件的帧缓冲区应当有下列特性。

自顶向下的格式，像素（0，0）在左上角，像素（width-1,height-1）在右下角。

帧缓冲区的步幅，即表示显示设备上一个扫描行在内存中占的字节数，应当是四字节的整数倍，即使这将意味着在每个扫描行的末尾将填充一些无用的字节。

CPU 不需做“排”选择就可以存取整个帧缓冲区。

帧缓冲区不应当使用位面，在位面中帧缓冲区被用于每种颜色信道或强度部件。

如果读者打算使用 GPE 类来实现一个显示设备驱动程序，但自己的显示设备硬件不支持 GPE 类（例如，如果帧缓冲区不是线性的），就可能会考虑编写一个“脏矩形”驱动程序。

在这种模式中，GPE 类维护一个在内存中的独立于设备的位图（DIB）来表示帧缓冲区。一旦在内存中的 DIB 被修改了，GPE 就会通知脏矩形驱动程序。脏矩形驱动程序就会响应，将 DIB 中改变了的或“脏”的部分复制到显示设备上，执行所需的任何改变。

脏矩形驱动程序在内存使用上和执行速度上花费很大的代价。这些应当仅仅作为实现对无法与 GPE 需要兼容的硬件进行支持的最后一种手段。

为了减少代价，微软推荐使用下列加速操作的显示设备硬件：

- 纯色填充。尤其是其 pbo->iSolidColor 成员不是 0xFFFFFFFF 的 Blt 操作。
- SRCCOPY Blt 操作。
- 如果你的平台使用光标，那么包括光标的显示。
- 用单像素以内的精确度绘制实线。
- 屏蔽 SRCCOPY Blt 操作。

7.2 Intel 的 IEGD 显卡驱动包

2005 年 4 月，在台湾举行的 Intel 开发者大会中，Intel 发布了专为机顶盒设计的“新芯片”。将超低电压版 Celeron M 处理器 854，使用 130 nm 生产技术，主频 600 MHz，400 MHz 系统总线。与 854 配套的芯片组最高支持 2 GB 333 MHz DDR SDRAM 内存，集成 Extreme Graphics 2 图像引擎。在网络连接方面支持 10/100 Mbps 宽带，6 个 PCI 插槽，支撑最高 6 个 USB 2.0 端口，所使用的南桥芯片是已经被 Intel 淘汰的 ICH4-M 类型。这一芯片组支持 Windows CE 5.0 操作系统，它有可能成为 Intel 机顶盒产品的主要操作系统。

在随后的几年中，Intel 的嵌入式产品线越来越丰富，性能也越来越高。目前，大量使用迅驰技术的 CPU 被应用于各种场合的嵌入式平台之中。相应地，各种嵌入式系统对软硬件的性能要求也越来越高。

如果将 WINCE6 作为一个控制系统，对显示性能没有太高的要求，也许读者不会碰到显卡驱动的问题。但是，对于大数据量的高速绘图，比如实时医学图像处理，显卡将成为性能的巨大瓶颈。

当然，WINCE6 的显示要求针对 PC 机上大型 3D 游戏来说，相差还是较大，但需要综合评估性能、价格等多项因素时，好的显卡与好的驱动能显著提高系统整体性能并大幅降低硬件成本。

WINCE6 的 BSP 附带的一个 VGA 显卡驱动，“VGA Linear （Flat） Framebuffer”，可以

驱动许多显卡，但该显卡驱动仅仅只能满足“显示”要求而已。

这一显示驱动实际上同 Loadcepc.exe 以及 BIOS Loader 的参数设置有关。

Loadcepc.exe 启动时，可以带一个“/L:”的参数，实际上这个是设置“VGA Linear （Flat） Framebuffer”显卡驱动的显示分辨率。其格式为：“Loadcepc /L:DXxDYxBPP”。如：Loadcepc /L:800x600x16。800 与 600 之间为小写字母 x。这些参数的配置可以在 DOS 系统的“AUTOEXEC.BAT”中进行修改。

相应地，BIOS BootLoader 配置完成之后，在硬盘上有一个“boot.ini”文件，其内容如下：

```
#
# BIOS loader configuation file
#

#
# Video settings
#

# Video on/off
Video=on

# Display screen width
DisplayWidth=800

# Display screen height
DisplayHeight=600

# Display screen depth
# Possible values: 8, 15, 16, 24, 32
DisplayDepth=32

# Physical screen width
# Must be greater or equal to DisplayWidth
# PhysicalWidth=640

# Physical screen height
# Must be greater or equal to DisplayHeight
# PhysicalHeight=480

#
# Debug settings
#

# Ethernet adapter base address (hex)
# EthIO=0x3f0

# Ethernet adapter IRQ number
# EthIRQ=5

# Debug IP address
# DbgIP=10.0.0.1
```

```
#
# Boot settings
#

# Primary BIN file name
BinFile=nk.bin

# Alternative image
# While booting hit the key corresponding to the suffix char (e.g. "1" for
# BinFile1) to boot the alternative image
# BinFile1=alt.bix

# Secondary BIN file name. Used when BinFile fails.
BakBinFile=eboot.bix

# Boot delay (in seconds) in which to select the alternative image
Delay=5

# Device name root string
DeviceNameRoot=CEPC

# Flash backup yes/no
FlashBackup=no

# Debug zone:
#    1 - error
#    2 - warning
#    4 - info
DebugZone=0x00000007

# COM Port to pass in BootArgs (Functionality must be supported by loaded program
[eboot/nk..etc] for argument to have an affect)
#    0 - Suppress
#    1 - COM 1
#    2 - COM 2
#    3 - COM 3
#    4 - COM 4
COMPort=1
```

其中 DisplayWidth 与 DisplayHeight 就是对“VGA Linear （Flat） Framebuffer”显示驱动分辨率的设置。

“VGA Linear（Flat）Framebuffer”显示驱动组件在 PB 中的位置如图 7-8 所示。

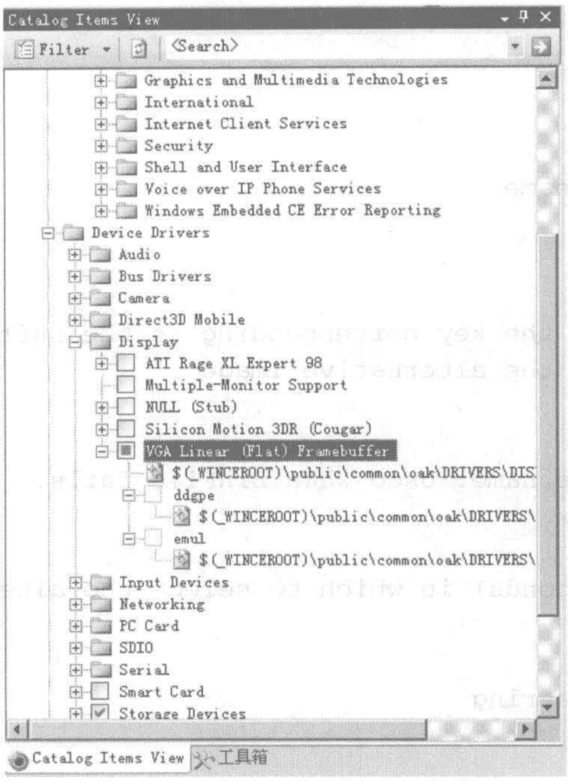


图 7-8

WINCE 的 BSP 本身自带的高性能显卡的驱动太少，而且可供选择的硬件平台限制过多，在工程应用上有诸多限制。

为此，Intel 提供了其公司当前主流显卡的高性能驱动程序，IEGD 显示驱动包。

下载地址：

http://downloadcenter.intel.com/Detail_Desc.aspx?agr=Y&ProductID=2159&DwnldID=16751&strOSs=103&OSFullName=Windows*%20CE%205.0&lang=zho

全新开发的英特尔嵌入式图形驱动程序（IEGD）特别针对嵌入式平台开发人员的需求，为台式机和移动式驱动程序市场提供备选产品。

IEGD 支持嵌入式英特尔架构路线图，包括英特尔 910GMLE、915GV、915GME、945G、945GME、Q965、GME965、Q35（无 OGL）、GM45（仅 2D）和 US15W 芯片组。此发行版是一个综合包，其中包含 IEGD 支持的所有操作系统版本以及预装配置编辑器（CED）实用程序。CED 实用程序用于在 Microsoft Windows 操作系统上运行。

此软件包内含 9.0.2 版驱动程序和用于在以下操作系统上运行的视频 BIOS：

IBM PC DOS 2000* 和 Microsoft DOS 6.22* 的视频 BIOS 支持

Microsoft Windows XP 专业版

Microsoft Windows XP Embedded

Microsoft Windows for Point of Service （WEPOS）

Microsoft Windows CE* 5.0

Microsoft Windows CE 6.0

以下分发版的 Linux* 支持：

——Damn Small Linux*

——Fedora Core 5*、6 和 7

——SUSE Enterprise Server （SLES） 10 和 10+ Linux

——Novel Linux Point of Service （NLPOS） 9 service pack 3

——Red Hat Embedded for US15W

——Red Flag Midinux for US15W

——Wind River Systems Embedded Linux for US15W

注：英特尔内核模块提供向几乎所有 Linux 分发版添加 IEGD 的代码示例。

如图 7-9 所示，IEGD 显示驱动程序安装过程按向导默认安装即可，安装目录为“C:\IEGD\IEGD_9_0_2”。

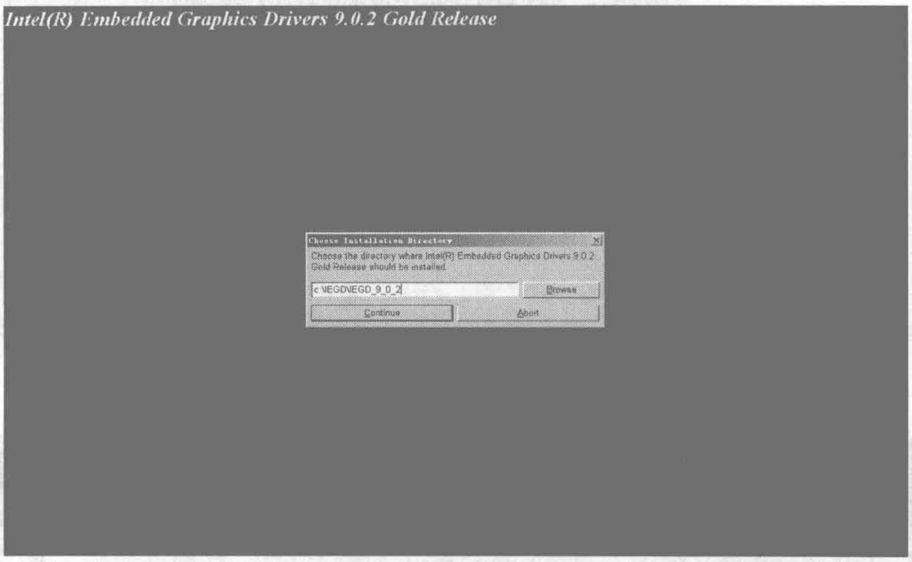


图 7-9

接下来介绍如何通过安装在 Windows 上的应用程序生成 WINCE6 的显卡驱动程序。

步骤 1 在安装目录下双击“iegd-ced.exe”或者通过快捷方式启动程序，启动界面如图 7-10 所示。

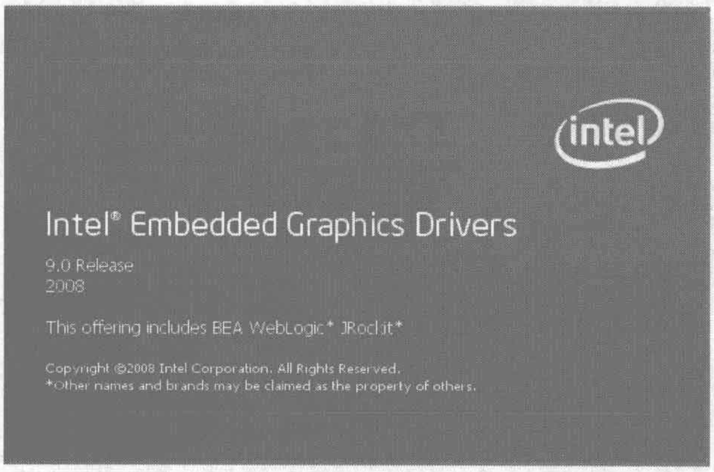


图 7-10

步骤 2 如图 7-11 所示，单击“New Configuration”按钮，在弹出界面中，如图 7-12 所示，填写“Configuration File Name”（可以随意填写），本例填写为“Test”，选择显示芯片组，

并设置显示分辨率与刷新率，选择显示器类型，然后单击“Next”按钮。

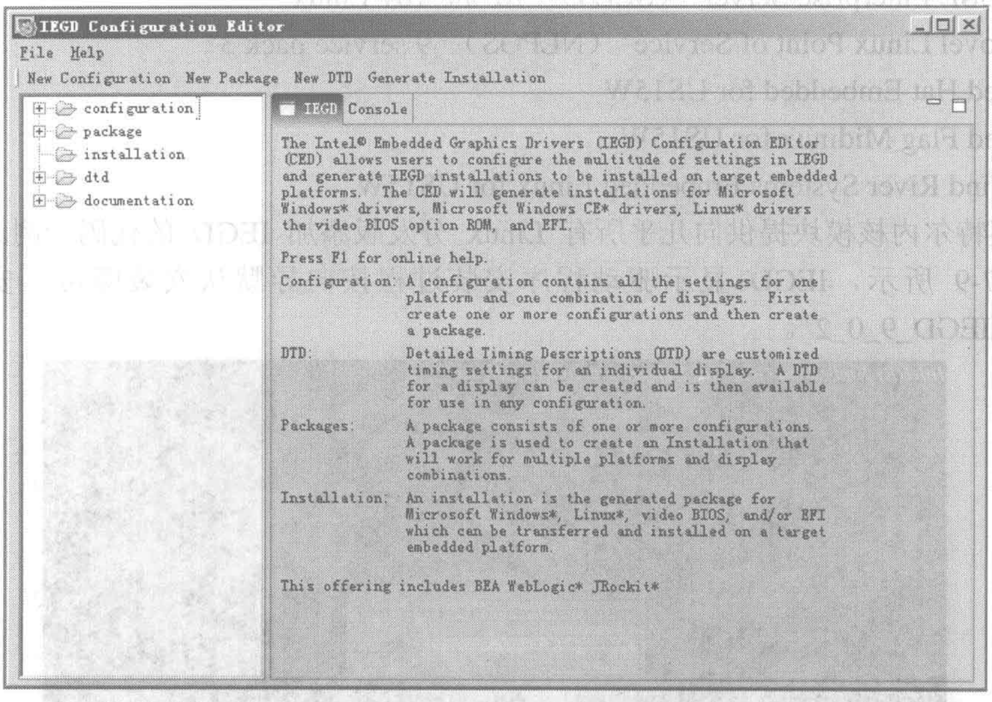


图 7-11

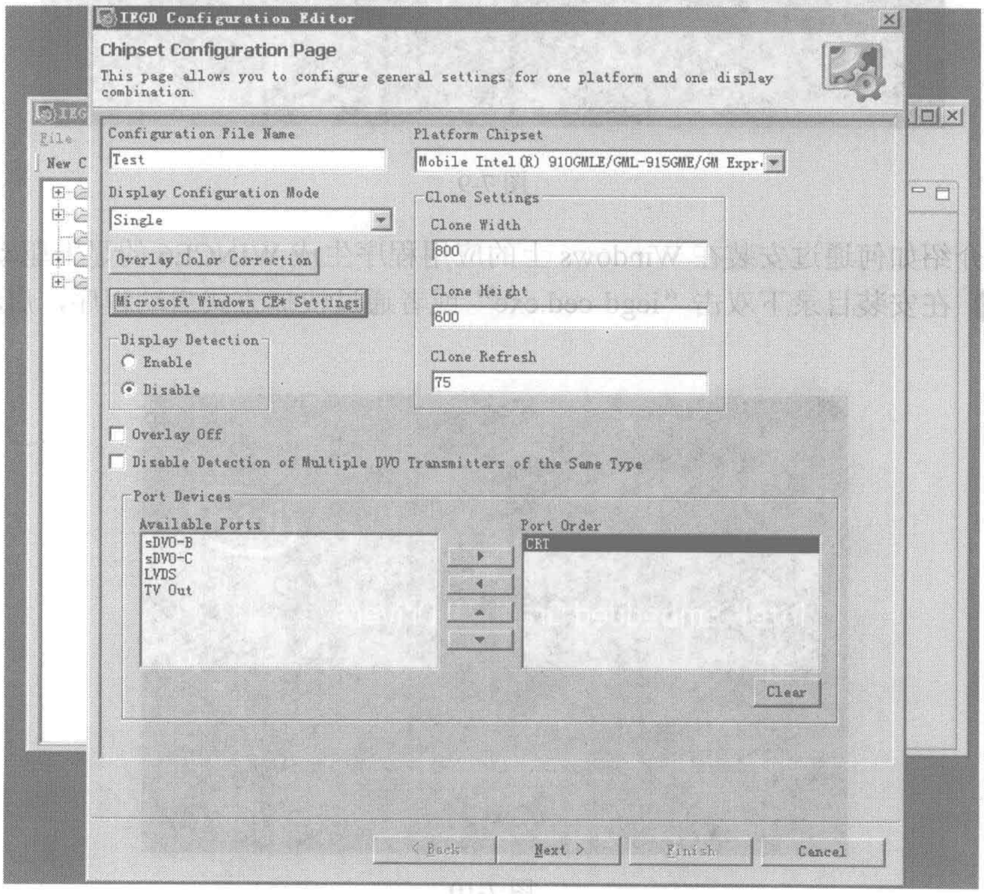


图 7-12

步骤 3 如图 7-13 所示，在新弹出的界面中填写“Readable Port Name”（可以随意填写），

本例填写为“Test”，然后单击“Finish”按钮。

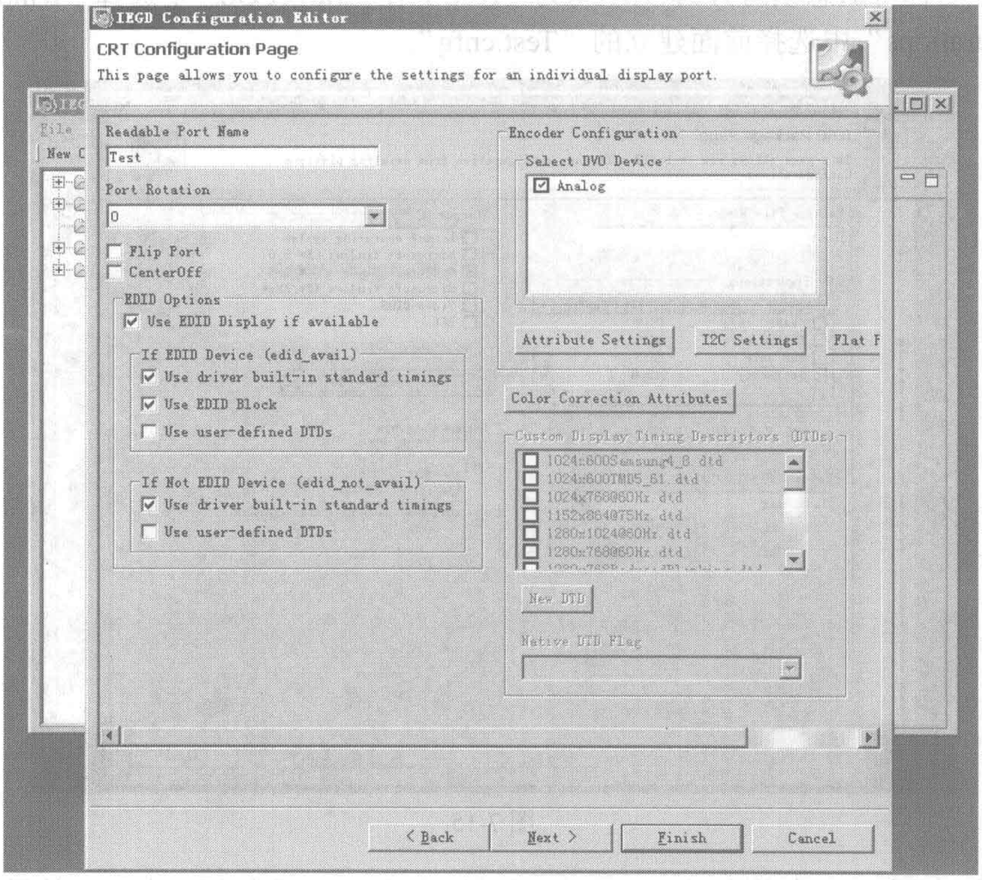


图 7-13

步骤 4 如图 7-14 所示，回到主界面，可以看到，在左侧的“Configuration”列表中增加了“Test.cnfg”。

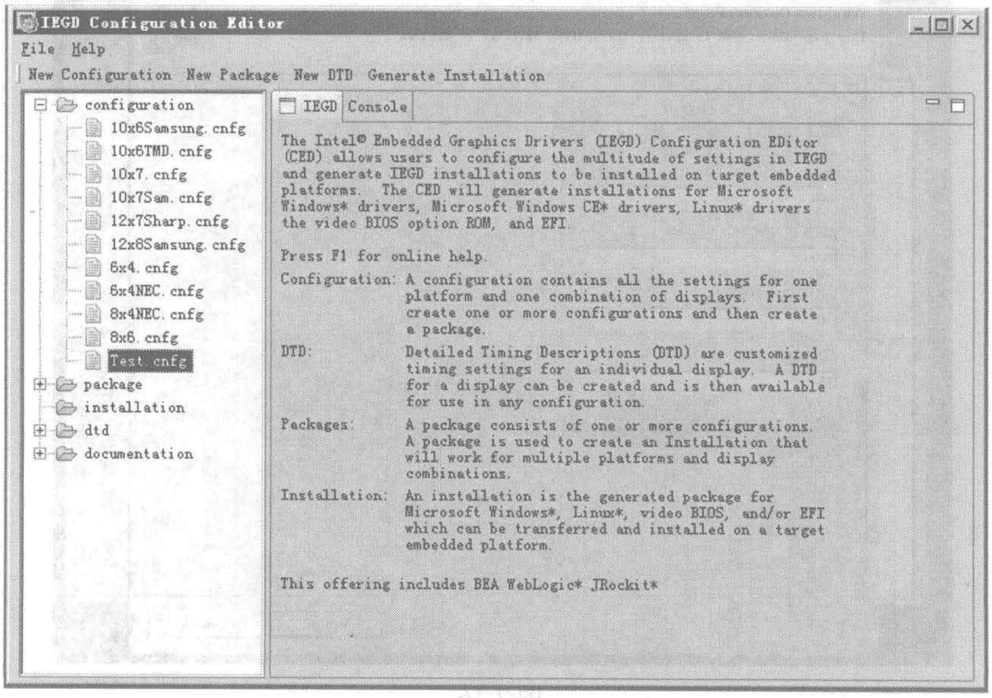


图 7-14

步骤 5 单击“New Package”按钮，在弹出的界面中（如图 7-15 所示），填写“Package File Name”（名称可随意填写），本例填写为“Test”，在“Target OS”中选择“Windows CE 6”，在“Configurations”中选择前面建立的“Test.cnfg”。

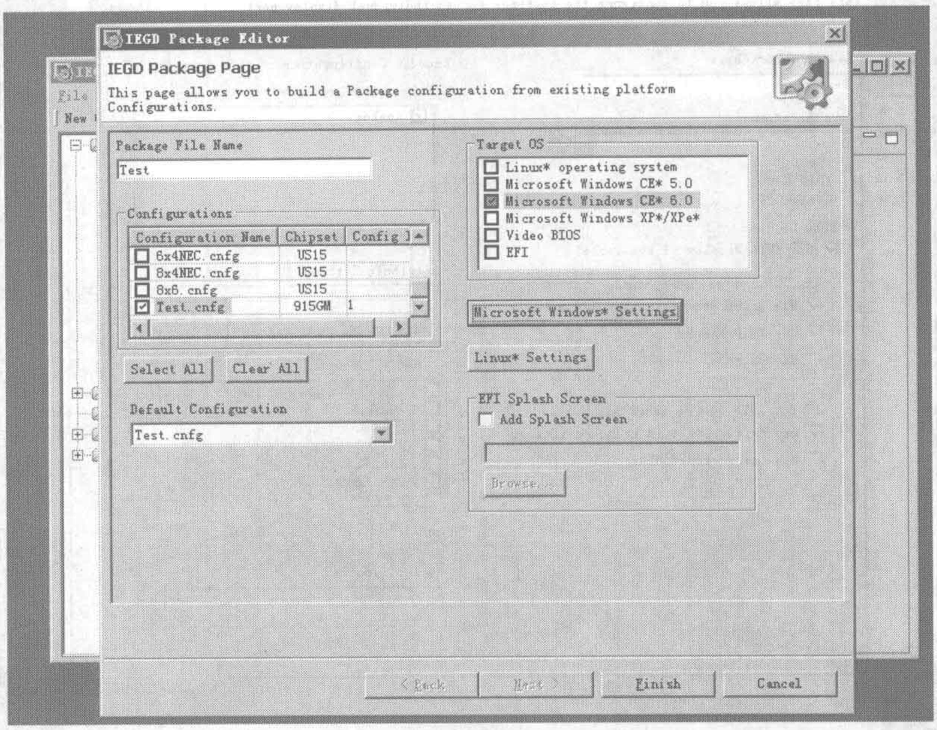


图 7-15

步骤 6 在图 7-15 中单击“Microsoft Windows* Settions”按钮，在弹出的界面中，如图 7-16 所示，填写显示分辨率与刷新率，然后单击“Finish”按钮，回到上层界面后再单击“Finish”按钮，回到主界面。

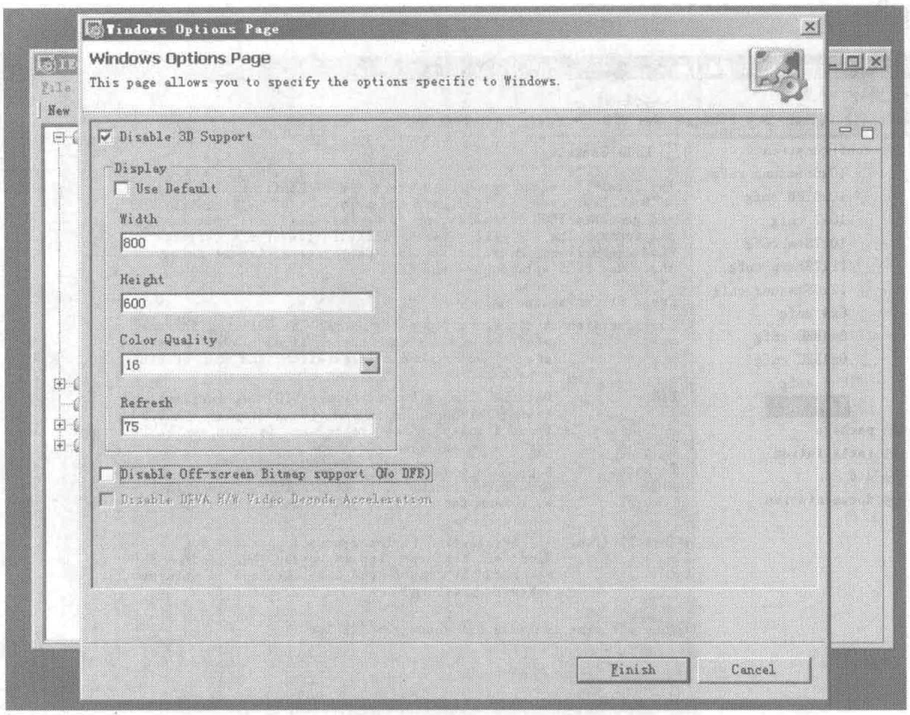


图 7-16

步骤 7 如图 7-17 所示，在主界面中的“Package”列表中选择新增加的“Test.pkg”，然后单击“Generate Installation”按钮，生成目标驱动。

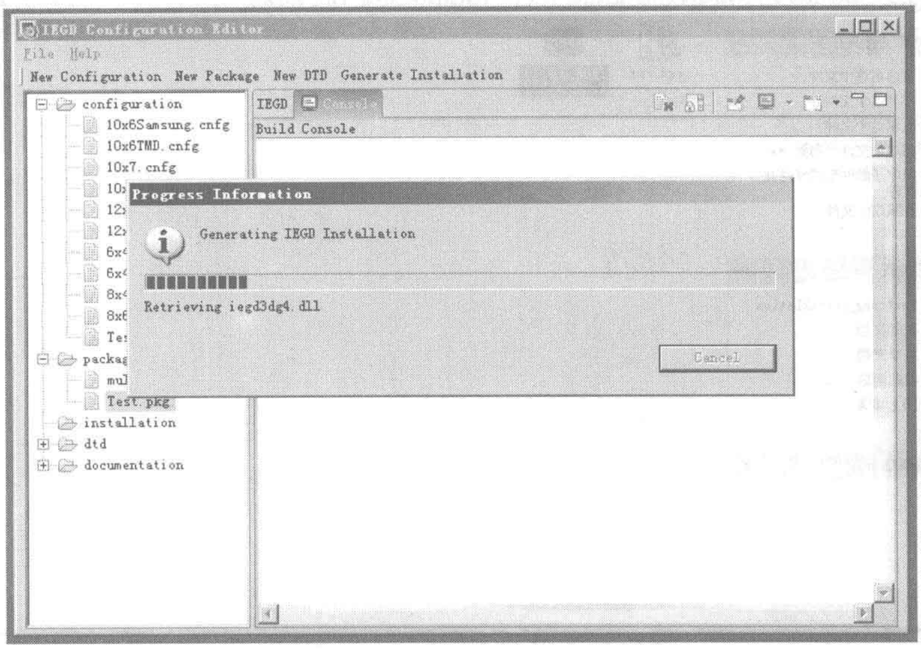


图 7-17

步骤 8 如图 7-18 所示，在主界面的“installation”列表中新增加了“Test.pkg_installation”，这就是最后生成的 WINCE6 的显卡驱动程序及注册表设置。

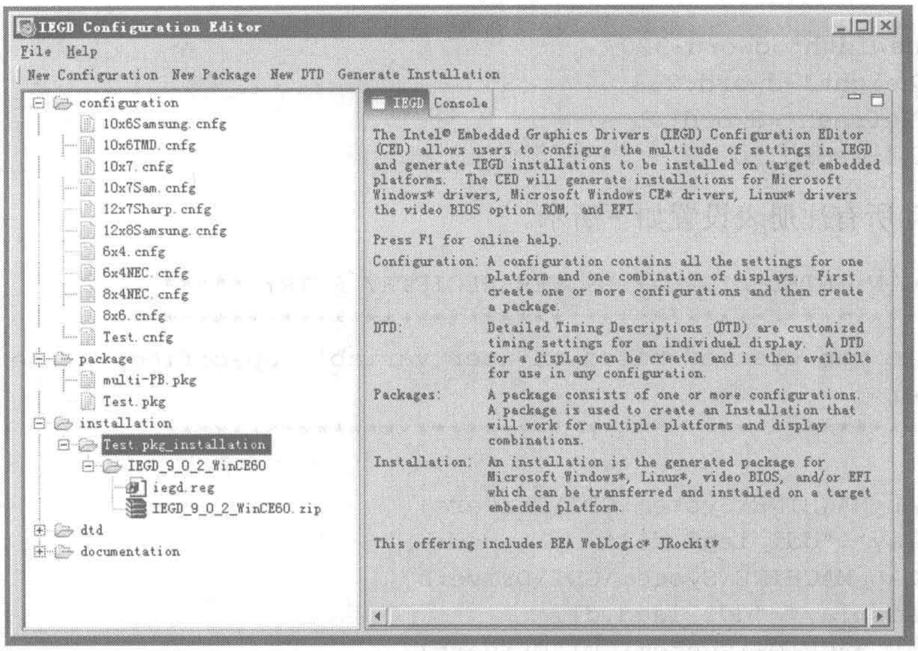


图 7-18

步骤 9 如图 7-19 所示，找到以下目录：“C:\IEGD\IEGD_9_0_2\workspace\installation\Test.pkg_installation\IEGD_9_0_2_WINCE60”。

将其中的“IEGD_9_0_2_WINCE60.zip”文件解压缩即可得到显卡驱动文件。

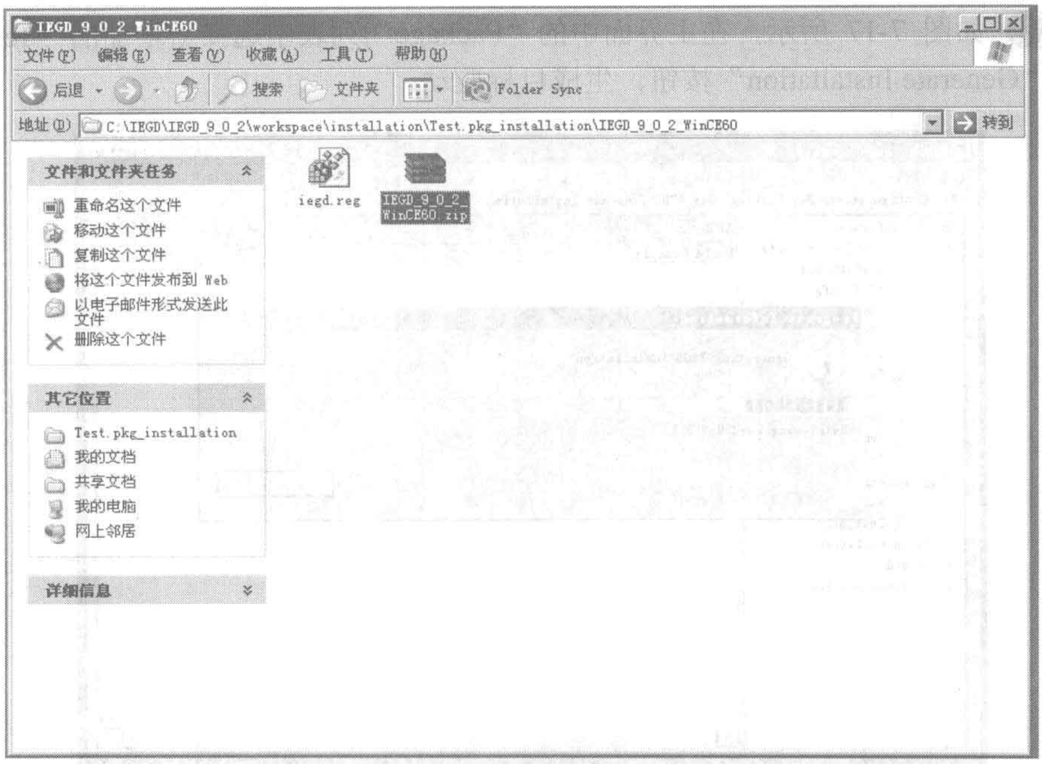


图 7-19

值得注意的是，与“VGA Linear（Flat）Framebuffer”显示驱动不同，该显示驱动程序设置分辨率与刷新率等参数是由注册表选项确定的，其值为 16 进制值。

如 800x600x32 的分辨率，75 Hz 的刷新率设置方式如下所示。

```
"CloneWidth"=dword:320
"CloneHeight"=dword:258
"CloneRefresh"=dword:4b
"Depth"=dword:20
```

该显示驱动所有注册表设置如下所示。

```
;***** BEGIN INTEL DISPLAY DRIVER REGISTRY ENTRY *****
;*****************************************************
; This file was created based on user variable specified in the CED.
; DriverVer=
;*****************************************************

[HKEY_LOCAL_MACHINE\System\GDI\Drivers]
    "Display"="ddi_iegd.dll"
;[HKEY_LOCAL_MACHINE\System\GDI\Drivers]
;    "MainDisplay"="ddi_iegd.dll"
[HKEY_LOCAL_MACHINE\System\GDI\Drivers]
   "D3DMOverride"="ddi_iegd.dll"
[HKEY_LOCAL_MACHINE\System\D3DM\Drivers]
   "RemoteHook"="ddi_iegd.dll"


;*****************************************************************
; The Following Sections Provide
```

```
; General Driver-Wide Registry Settings
;*******************************************************************
[HKEY_LOCAL_MACHINE\Drivers\Display\Intel]
   "PcfVersion"=dword:700
   "ConfigId"=dword:1

;*******************************************************************
; The Following Sections Provide Per-Config
; & Per-Port Registry Settings
;*******************************************************************

[HKEY_LOCAL_MACHINE\Drivers\Display\Intel\ALL\1]
   "name"="Test"
[HKEY_LOCAL_MACHINE\Drivers\Display\Intel\ALL\1\General]
;-------------------------------------
; Select Display configuration, single, twin ...
;-------------------------------------
   "DisplayConfig"=dword:1
;-------------------------------------
; Select if you want to enable Display Detection
;-------------------------------------
  "DisplayDetect"=dword:0
;-------------------------------------
; Select Port Order
;-------------------------------------
   "PortOrder"="50000"
   "CloneWidth"=dword:320
   "CloneHeight"=dword:258
   "CloneRefresh"=dword:4b
   "Depth"=dword:20
   "MaxFbSize"=dword:800000
   "MinVidSurfX"=dword:10
   "MinVidSurfY"=dword:10
   "SysToVidStretch"=dword:1
   "BlendFilter"=dword:0
   "TearFB"=dword:1
   "NO_D3D"=dword:0
   ;-------------------------------------
   ; Config 1 - Test Port
   ;-------------------------------------
   ; Following are the registry
   ; entries for port's general config
   ;-----------------------------
   [HKEY_LOCAL_MACHINE\Drivers\Display\Intel\ALL\1\Port\5\General]
      "Name"="Test"
      "Rotation"=dword:0
      "Edid"=dword:1
      "EdidAvail"=dword:3    ; STD TIMINGS + EDID TIMINGS + USER TIMINGS
      "EdidNotAvail"=dword:1 ; STD TIMINGS + USER TIMINGS
   ;-----------------------------
   ; Following are the registry entries
   ; for port's DVO I2C settings
```

```
    ;--------------------------------
    [HKEY_LOCAL_MACHINE\Drivers\Display\Intel\ALL\1\Port\5\DVO]
    ;--------------------------------
    ; Following are the registry entries
    ; for port's flat panel's mode-limits,
    ; power and backlight control
    ;--------------------------------
    [HKEY_LOCAL_MACHINE\Drivers\Display\Intel\ALL\1\Port\5\FPInfo]
    ;--------------------------------
    ; Following are the registry entries
    ; for ports first custom DTD mode to add
    ;--------------------------------

    ; Following are the registry entries for the port device' display attribute
parameters
    ;
    ;--------------------------------
    [HKEY_LOCAL_MACHINE\Drivers\Display\Intel\ALL\1\Port\5\Attr]

  ;--------------------------------------
  ; Provide a list of port drivers to attempt to load upon boot time
  ;--------------------------------------
  [HKEY_LOCAL_MACHINE\Drivers\Display\Intel]
    "PortDrivers"="analog"
  ;--------------------------------------
  ; Optional - Only enable for font anti-aliasing
  ; Enabling this causes minor performance impact
  ; Only recommended for TV Output.
  [HKEY_LOCAL_MACHINE\System\GDI\Fontsmoothing]

  [HKEY_LOCAL_MACHINE\System\GDI]
    "ForceGRAY16"=dword:1

  ;***** INTEL DISPLAY DRIVER REGISTRY ENTRY END *******
  ;****************************************************
```

该驱动程序的说明如下所示（以下摘自该显示驱动程序的“**Release Note**”，为确保其原意故不作翻译）。

```
==========================================================================
Release Notes for the
Intel(R) Embedded Graphics Drivers (IEGD)
with Configuration Editor (CED) Package

Version 9.0 Beta Release
May, 2008
==========================================================================

Contents
========

- In This Release (overview)
```

```
- Supported Chipsets
- New Features
- Getting Started
- VBIOS
- Linux
- Windows*
- Windows CE 5.0 / 6.0
- Software Errata
- Software Licenses
- Legal / Disclaimers


In This Release
================

This is a release of the Intel(R) Embedded Graphics Drivers (IEGD).
IEGD is designed specifically to support the unique GRAPHICS DRIVER
requirements of embedded applications for Intel embedded chipset-based
platforms with integrated graphics capability.

This release is a comprehensive package that includes all supported
O/S versions of IEGD, and the pre-installation configuration editor
(CED) utility.  Highlights of the included files/folders are:

iegd-ced.exe                              Configuration EDitor main file
License.txt                               IEGD License

\configuration                            CED main files

\features                                 CED copyright, license, and
                                          version info

\JRE                                      Java engine for CED

\Plugins                                  CED program files and support
                                          files
\plugins\com.intel.iegd.drivers.xxx       Individual IEGD drivers

\workspace                                Configured drivers and
                                          associated files
\workspace\Documentation\RELNOTES.TXT     Release Notes
\workspace\Documentation\UsersGuide.pdf   IEGD User's Guide


Supported Chipsets
==================

Target system must contain one of the following Intel Chipsets:

           Intel(r) US15 System Controller Hub (Poulsbo)
     Mobile Intel(R) GM45/GL40/GS45 Express Chipset (Cantiga)
           Intel(R) Q35 Express Chipset
```

```
        Mobile Intel(R) GLE960 Express Chipset
        Mobile Intel(R) GME965 Express Chipset
               Intel(R) Q965 Express Chipset
        Mobile Intel(R) 945GME Express Chipset
               Intel(R) 945G Express Chipset
        Mobile Intel(R) 915GME Express Chipset
               Intel(R) 915GV Express Chipset
        Mobile Intel(R) 910GMLE Express Chipset

Please check with your system provider to determine the Intel Chipset
used in your system.


New Features
============

This release contains the following new features and capabilities:

Added:
    - Intel(R) US15 System Controller Hub (2D only)
    - Intel(R) GM45/GL40/GS45 Express chipset support (2D only)
    - Intel(R) GLE960 Express chipset support
    - HDMI 1.2a and HDCP support (driver only)
    - Linux: Red Flag MIDINUX 2.0
    - Linux: Red Hat Embedded Linux
    - Linux: Wind River Linux
    - Linux ACPI

Removed (no longer supported):
    - Intel(R) 855GME Chipset (Mobile), Intel(R) 852GM Chipset (Mobile)
      and Intel(R) 852GME Chipset (Mobile) are no longer supported.


This release also contains fixes for errata.

For details on errata, including status information, refer to the Errata
document
usually located at the following Web address:

 www.intel.com/design/intarch/specupdt/309380.htm

or available to download with this release.


Getting Started
===============

The first step is to install the CED release. The setup and CED utility
currently need a Windows (XP, XPe, etc.) environment in which to operate.
Simply click on the release to run the setup. Default directory is:

 \IEGD\IEGD_x_x_x
```

where x_x_x is the version number and release type (Gold, hot-fix, beta, EC, etc.)

New versions will install in their own directory and do not disturb any existing CED installations so you can switch between versions as needed.

Once installed, you need only switch to the folder and click on the file:

iegd-ced.exe

which will run the CED utility.

NOTE: Only one iteration of CED should be run at any time and as such the program uses a semaphore lock file to prevent multiple versions from being run at the same time. If you receive an error indicating that another instance of CED is running yet no other copies are running, then you can clear the improper lock by going into the WORKSPACE folder in the version of CED you are attempting to run and delete the file named ".lock" which will clear the lock.

Please see the IEGD User's Guide for your release for detailed configuration instructions.

=====
VBIOS
=====

VBIOS System Requirements
=========================

The VBIOS can be built on a host system and moved to the target system. The host system should have installed a 32-bit Windows operating system with the capability to execute DOS commands from a command line window.

The Target system should have DOS installed. Supported DOS versions are MS-DOS* 6.22, MS-DOS 7.1 (from Windows 98), or IBM PC DOS 2000*.

The Target system should contain a minimum of 64MB of RAM.

VBIOS Files
===========

The individual install file (for manual access/configuration) is found under the IEGD install directory in \plugins\com.intel.iegd.drivers.xxx\vbios folder (where xxx represents the release version)

The following describes the files used for building the VBIOS for each supported platform, where zzz indicates the chipset for which to build the

```
   specific Option ROM or TSR.

   /Driver/zzz/orom/igd_uinit.h    Header file containing default
                                     initialization settings
   /Driver/zzz/orom/makefile       Makefile used to build VBIOS Option ROM
   /Driver/zzz/orom/pd_init_all.c  C file containing structures and default
                                     initialization settings
   /Driver/zzz/orom/vga-def.bin    Default VBIOS Option ROM for single CRT
                                     display
   /Drivers/zzz/orom/lib/*         Library files used to build the VBIOS
                                     option ROM
       libigdpialm.lib             For Almador graphics engine based chipsets
                                     (852/855)
       libigdpinap.lib             For Napa graphics engine based chipsets
                                     (915/945)
       libigdpign4.lib             For Gen 4 graphics based chipsets
                                     (965)

   /Driver/zzz/tsr/igd_uinit.h     Header file containing default
initialization
                                     settings
   /Driver/zzz/tsr/makefile        Makefile used to build VBIOS TSR
   /Driver/zzz/tsr/pd_init_all.c   C file containing structures and default
                                     initialization settings
   /Driver/zzz/tsr/tsr-def.bin     Default VBIOS TSR for single CRT display
   /Driver/zzz/tsr/lib/*           Library files used to build the VBIOS TSR

   /Utilities                      Folder containing DOS utilities that
interface
                                      with the Video BIOS
   /Utilities/readedid.exe         Utility for retrieving panel EDID information
   /Utilities/readmode.exe         Utility for retrieving current display mode
   /Utilities/setmode.exe          Utility for setting a display mode
   /Utilities/setpipe.exe          Utility for setting active display pipe
   /Utilities/vbebat02.exe         VESA mode acceptance test, for diagnostic
purposes
   /Utilities/vga00.exe        Basic mode diagnostic test


   VBIOS: Building the VBIOS
   =========================

   The Intel Embedded VBIOS should be built using the CED utility. The VBIOS
   is built based on settings in CED configuration. When directed to do so,
   CED will automatically build both the VGA.BIN ROM image and a TSR.EXE based
   on your CED settings.  These files are ready to use directly out of the CED.
   Output will be found under the \workspace folder based on the chipset(s)
   you built. Details are provided in the User's Guide and
   in the CED online help.

   The Intel Embedded VBIOS may no longer be build manually using command line
```

utilities.

VBIOS: Build Prerequisites

Before building your VBIOS in the CED, you must set up your environment with the Open Watcom 16-bit compiler:

The CED relies on the Open Watcom C/C++ compiler to be able to build a 16-bit DOS binary required for the BIOS. The VBIOS has been tested with version 1.3 of the Open Watcom compiler. The latest version can be downloaded from the following location:

http://www.openwatcom.com

You must set up the PATH in DOS to be able to execute the Watcom compiler. The directories for the ml.exe and nmake.exe programs should be included in the execution path variable in your DOS environment.

VBIOS: TSR and Option ROM

The TSR.exe (Terminate and Stay Resident) executable is useful for testing you VBIOS build before merging the option ROM with your system firmware. This program can be copied to the target machine. To run the TSR, boot the target machine with DOS, and then run the TSR.exe from the DOS command line.

The CED also creates a VGA.bin file, which is the Option ROM which must be merged with your system firmware. Contact your firmware provider for instructions on merging option ROMs with the system firmware.

End VBIOS
****

=====
Linux
=====

Linux: System Requirements
==========================

This package includes drivers built for the following X servers:

```
  XFree86 version 4.3.0
  X.org version 6.7, 6.8, 7.0, 7.1, & 7.2

Note the driver requires the official versions from www.Xfree86.org
or www.X.org, and may not necessarily operate with versions that are
part of your Linux distribution.

The system should contain a minimum of 128MB of RAM.

The Linux kernel Accelerated Graphics Port Graphics Address Remapping Table
(AGPGART) must support the supported Intel chipsets (see
below for more information).


Linux: Files
============

The individual Linux install file (for manual access/configuration) is found
under the IEGD install directory in \plugins\com.intel.iegd.drivers.xxx\linux
folder (where xxx represents the release version).

This package contains an XFree86 loadable module device driver for the
Intel chipsets supported by this release.

The following files are contained in the Linux release:

License/License.txt                    License file for display driver
Documents/RELNOTES.txt                 Release notes
Documents/UsersGuide.pdf               IEGD User's Guide
Documents/Xfree86/intel.4              Man page for display driver -
Xfree86
Documents/Xfree86/IntelEscape.3x       Intel(R) Escape man page -
Xfree86
Documents/Xorg-X11/intel.4             Man page for display driver -
X.org
Documents/Xorg-X11/IntelEscape.3x      Intel(R) Escape man page - X.org

Driver/XFree86-4.3/iegd_drv.o          Intel(R) Embedded Graphics
Driver binary
Driver/XFree86-4.3/libGLgn3.so         Intel(R) Embedded Graphics
Driver OpenGL
                                        library for 945 and earlier
chipsets
Driver/XFree86-4.3/libGLgn4.so         Intel(R) Embedded Graphics
Driver OpenGL
                                        library for 965 chipsets
Driver/XFree86-4.3/libXiegd_escape.a   Intel(R) Escape client library.
Driver/XFree86-4.3/libXiegd_escape.so.2.0.0 Intel(R) Escape client shared
library
Driver/XFree86-4.3/intel_escape.h      Intel(R) Escape client include
Driver/XFree86-4.3/ch7009.so           Chrontel CH7009/7301 TVout/DVI
encoder
```

```
   Driver/XFree86-4.3/ch7017.so              Chrontel CH7017/7019/7304/7305*
TVout/LVDS
                                             encoder
   Driver/XFree86-4.3/fs454.so               Focus FS453/454 TVout encoder
   Driver/XFree86-4.3/lvds.so                GMCH Internal LVDS controller
   Driver/XFree86-4.3/tv.so                  GMCH Internal TV controller
   Driver/XFree86-4.3/ns2501.so              National Semiconductor NS2501
LVDS
                                             controller
   Driver/XFree86-4.3/ns387.so               National Semiconductor NS387R
LVDS
                                             controller
   Driver/XFree86-4.3/sii164.so              Silicon Image SiI164 DVI encoder
   Driver/XFree86-4.3/ti410.so               Texas Instruments TFP410 DVI
encoder
   Driver/XFree86-4.3/th164.so               Thine 164 DVI encoder
   Driver/XFree86-4.3/sdvo.so                Serial DVO Devices
   Driver/XFree86-4.3/softpd.so              VCH Devices

   Driver/Xorg-X11R6.7/*.*                   Xorg-X11R6.7 binaries

   Driver/Xorg-X11R6.8/*.*                   Xorg-X11R6.8 binaries

   Driver/Xorg-X11R7.0/*.*                   Xorg-X11R7.0 binaries

   Driver/Xorg-X11R7.1/*.*                   Xorg-X11R7.1 binaries

   Driver/Xorg-X11R7.2/*.*                   Xorg-X11R7.2 binaries

   Utilities/gui_lib_setup_linux.txt         Instructions for GUI setup
   Utilities/iegdgui                         Display modes GUI utility
   Utilities/iegd_crg                        Customer Reprt Generator for
reporting
                                              driver issues

   The following files are contained in the IEGD_Patches directories:

   IEGD_Patches/IKM                IEGD Kernel Module for GART and DRM
                                     Replaces the patches for kernels 2.6 and later

   IEGD_Patches/DRIVER             Kernel patches for AGP GART

   agpgart.patch-2.4.20            Patch for 2.4.20 kernel
   agpgart.patch-2.4.20-8          Patch for 2.4.20-8 kernel (Red Hat* 9)
   agpgart.patch-2.4.24            Patch for 2.4.24 kernel
   agpgart.patch-2.4.26            Patch for 2.4.26 kernel
   agpgart.patch-2.6.5-1.358       Patch for 2.6.5-1.358 kernel (Fedora* 2)
   agpgart.patch-2.6.5-7.191       Patch for 2.6.5-7.191 kernel (older SUSE)
   agpgart.patch-2.6.5-7.244       Patch for 2.6.5-7.244 kernel (NLPOS 9 SP3)
   agpgart.patch-2.6.13-15         Patch for 2.6.13-15 kernel (SUSE 10)
   agpgart.patch-2.6.15-1.2054     patch for 2.6.15-1.2054 kernel (Fedora 5)
```

```
agpgart.patch-2.6.16.46-0.12  patch for 2.6.16.46-0.12 kernel (SUSE 10 SP1)
agpgart.patch-2.6.18-1.2798   Patch for 2.6.18-1.2798 kernel (Fedora 6)
agpgart.patch-2.6.21-1.3194   Patch for 2.6.21-1.3194 kernel (Fedora 7)

IEGD_Patches/DRM              DRM files for OPenGL support

iegd.h
iegd_drm.h
iegd_drm_client.h
iegd_drv.c
iegd_drv.h
iegd_interface.c
iegd_interface-2611.c
iegd_interface-2611.h
iegd_interface-2615.c
iegd_interface-2615.h
iegd_interface-265.c
iegd_interface-265.h
Makefile                      Makefile to build the DRM


Linux: Installation
===================

Please see the IEGD User's Guide for detailed installation instructions.

Note: Default installation locations can vary so be sure to check your
system for the proper path.

Linux: Patch /IKM Information
-----------------------------

Additional kernel changes are needed for the GART driver to support the
Intel(R) Embedded Graphics X display driver.  This may be in the form of
a patch, or may be a new GART interface such as the IEGD Kernel Module (IKM)
which provides the necessary GART and DRM interfaces for IEGD.

Please see the User's Guide for details.


OpenGL*
-------

The OpenGL implementation for IEGD consists of three components.

    libGL.so: This is the shared library that implements the OpenGL and
        GLX APIs. It is linked by all OpenGL applications.
```

```
    intel.ko: This is the Direct Rendering Manager (DRM). It is a kernel
      module that provides the OpenGL application with the
      permissions necessary to directly access the DMA buffers
      used by libGL.

    X Server: The existing IEGD X server driver has been enhanced to
      communicate with libGL.

Installing the IEGD OpenGL driver provides a fully hardware accelerated
implementation of the OpenGL library to applications. This implementation
makes use of a direct rendering technology, which allows the client to
directly write to DMA buffers that are used by the Graphics Hardware.

Due to the use of direct rendering technology, system designers should
take special care to ensure that only trusted clients are allowed to use
the OpenGL library. A malicious application could otherwise use direct
rendering to destabilized the graphics hardware or, in theory, elevate
their permissions on the system.

A system designer can control the access to the direct rendering functionality
by limiting the access to the DRI device file located at:

   /dev/dri/card0

The permissions on this device are set by the X-Server using the information
provided in the "DRI" section of the xorg.conf file.

OGL Install:
------------
Please see the detailed install instructions in the Linux section of the
User's Guide to install the IEGD libGL

    NOTE: The system likely has a version of libGL already installed.
    You may want to make a backup copy of the existing library before
      installing the IEGD library.

End Linux
*********


=======
Windows
=======

Windows System Requirements
===========================

The development system should have one the following Windows operating
systems installed.

 Microsoft Windows XP*                SP2
```

```
 Microsoft Windows XP Professional*    SP2
 Microsoft Windows XP Embedded*        SP2
 Microsoft WEPOS*

The system should contain a minimum of 128MB of RAM.


Windows Files
=============

The individual install file (for manual access/configuration) is found
under the IEGD install directory in
\plugins\com.intel.iegd.drivers.xxx\windows
folder (where xxx represents the release version)

The following are found in the Windows Installation:

Driver\iegdmini.sys            Display driver
Driver\iegddis.dll             Hi level Interface Dynamic Link Library
Driver\iegd3dgn3.dll           Direct3D support Dynamic Link Library
                                for 945 and earlier chipsets
Driver\iegd3dgn4.dll           Direct3D support Dynamic Link Library
                                for 965 chipsets
Driver\iegd.inf                Install file for Display driver
Driver\iegd.sld                Sample Windows XP* Embedded Component
definition
                                file - you will need to build this from your
                                specific iegd.inf for your installation!
Driver\ch7009.sys              Port driver for Chrontel* CH7009/CH7301 encoder
Driver\ch7017.sys              Port driver for Chrontel CH7017/CH7305 encoder
Driver\fs454.sys               Port driver for Focus 453/454 TVOut encoder
Driver\lvds.sys                Port driver for Internal (GMCH) LVDS controller
Driver\tv.sys                  Port driver for Internal (GMCH) TV controller
Driver\ns2501.sys              Port driver for National Semiconductor* 2501
LVDS
                                controller
Driver\ns387.sys               Port driver for National Semiconductor 387R
LVDS
                                controller
Driver\th164.sys               Port driver for Thine* 164 encoder
Driver\sii164.sys              Port driver for Silicon Image* 164 encoder
Driver\sdvo.sys                Port driver for SDVO devices
Driver\ti410.sys               Port driver for Texas Instruments* TFP410
encoder
Driver\softpd.sys              Port driver for VCH devices
Driver\License.txt             License governing the use of the driver-
                                Please READ and agree before using the driver!
Utilities\IEGDGUI.dll          Library file for display modes GUI
Utilities\IEGDGUI.exe          Stand-alone display modes GUI executable
Utilities\iegd_crg.exe         Customer Reprt Generator for reporting driver
issues
```

```
Utilities\Setup.exe            Install/uninstall utility for driver
Utilities\display_license.txt License agreement for display driver


Installation and De-installation
================================

Details can be found in the Windows section of the IEGD User's
Guide.

This release contains an installer/uninstaller utility.  This utility
does not, however, configure the driver.  Please be sure to update your
ssigd.inf file with appropriate settings for your display configuration
prior to installation.  The installation program requires specific file
names. If you have a different INF file for driver installation, you
must name it iegd.inf and place it in the Drivers folder prior to
running the installation with SETUP.EXE.

Do not move or alter the directories after unzipping the release
package.  The installation utility uses relative path information, and
must be run from the Utilities folder, and also requires the Driver
folder.  If the folders are moved or renamed, the installation utility
will not operate properly.

Also, be sure to not install this version of the driver with an
existing installation of a previous version of the driver.  The
installation utility will not uninstall previous versions of the Intel
Embedded Graphics Driver, and you must begin from a fresh operating
system installation.  Results are unpredictable if you install over a
previous version of the Intel Embedded Graphics Driver.

To run the installation, simply run the Setup.exe program located in
the Utilities folder of the installation.  This must run from the
Utilities folder.

To uninstall this version of Intel Embedded Graphics Driver, run the
Setup.exe program.


End Windows
***********


====================
Windows CE 5.0 / 6.0
====================
```

```
WCE: System Requirements
============================

Development systems should have the following software
installed.

 Platform Builder for Windows CE (latest service packs)
     For WCE 6.0: VS2005 (latest)

The target system should contain a minimum of 64MB of RAM, or 128MB if you
will be using D3DM.

WCE: Files
====================================

The individual install file (for manual access/configuration) is found
under the IEGD install directory in \plugins\com.intel.iegd.drivers.xxx\WCE50
or WCE60 folders (where xxx represents the release version)

This package contains the following drivers, files, and documents.

Documents\RELNOTES.TXT        Release Notes
Documents\UsersGuide.pdf      IEGD User's Guide
License\License.txt           License agreement for display driver
Driver\DDI_IEGD.DLL           Display driver binary
Driver\DDI_IEGD.REL           Compilation REL file
Driver\IEGD.reg               Registry settings for the platform
Driver\config.bib             Example memory config file
Driver\igd_Wince50.cec        WCE 5.0 Catalog feature file (not in WCE6.0
package)
Driver\ch7009.dll             Port driver for Chrontel CH7009/CH7301 encoder
Driver\ch7009.rel             File required for Chrontel CH7009/CH7301 build
process
Driver\ch7017.dll             Port driver for Chrontel CH7017/CH7305 encoder
Driver\ch7017.rel             File required for Chrontel CH7017/CH7305 build
process
Driver\fs454.dll              Port driver for Focus 453/454 TVOut encoder
Driver\fs454.rel              File required for Focus 453/454 build process
Driver\lvds.dll               Port driver for Internal (GMCH) LVDS controller
Driver\lvds.rel               File required for Internal (GMCH) LVDS build
process
Driver\tv.dll                 Port driver for Internal (GMCH) TV controller
Driver\tv.rel                 File required for Internal (GMCH) TV build
process
Driver\ns2501.dll             Port driver for National Semiconductor* 2501
LVDS
                              controller
Driver\ns2501.rel             File required for National Semiconductor 2501
LVDS
                              build process
Driver\ns387.dll              Port driver for National 387R LVDS controller
```

```
   Driver\ns387.rel             File required for National Semiconductor 387R
LVDS
                                build process
   Driver\ti410.dll             Port driver for Texas Instruments* TFP410
encoder
   Driver\ti410.rel             File required for TFP410 encoder build process
   Driver\th164.dll             Port driver for Thine* 164 encoder
   Driver\th164.rel             File required for Thine 164 build process
   Driver\sii164.dll            Port driver for Silicon Image* 164 encoder
   Driver\sii164.rel            File required for Silicon Image 164 build process
   Driver\sdvo.dll              Port driver for SDVO devices
   Driver\sdvo.rel              File required for SDVO encoder build process
   Driver\softpd.dll            Port driver for VCH devices
   Driver\softpd.rel            File required for VCH encoder build process

   WCE: Functions Supported
   ============================
   1. Mode setting
   2. Hardware cursor
   3. GDI Line Acceleration
   4. Mode setting thru registry
   5. 8, 16, and 32 bit color modes
   6. All supported modes available
   7. GDI blitting acceleration (colorfill, stretched, source)
   8. DirectDraw support (overlay, blitting, flipping, etc)
   9. Text Anti-alias
   10. DVO port support
   11. ACSFL support
   12. Attribute control
   13. Twin, Clone and Vertical Extended modes
   14. Optional sys-to-video stretch blit
   15. Blend filter options
   16. Dynamic Video Memory Technology
   17. Minimum dimension video surface allocation rule
   18. NOTEARING support
   19. Loadable port driver architecture
   20. Per port Custom DTD support.
   21. Per port GPIO I2C configuration.
   22. Per port FP Native Resolution, Power & Backlight controls.
   23. Per port Attribute initialization via registry
   24. D3DM support
   25. Second overlay in vertical extended mode

   WCE: Installation
   =====================

   Details can be found in the Windows CE section of the IEGD User's
   Guide.

   Integration/installation of the driver binaries depends upon the requirements
```

```
of
   the target project workspace.

   If taken directly from this release package, the driver binaries need to be
   manually integrated into the project workspace on Platform Builder.

   In your current project workspace, remove the default display device driver.
Add
   a new user feature to the project and browse for the extracted ddi_igd.dll file.
   When rebuilding your project, Platform Builder will request to select a
   replacement display device driver to solve dependencies.  Select the
NULL(Stub)
   display device driver listed from the menu of choices offered.

   To install a DVO device port driver, use the platform settings menu and add
the
   desired port driver DLL into the BIB image.

   The contents of the extracted IEGD.reg file must be copied into the project
   registry file (project.reg) every time for each new project, or into the
   platform registry file (platform.reg) if all following projects on your current
   platform will run on the Intel graphics chipset.

   Please see the latest version of the IEGD User's Guide for more details.

   WCE: Catalogue Feature File
   ===============================

   A Catalogue Feature File has been included for WCE 5.0. In order to
   add this driver into the Platform Builder's catalogue feature list, complete
the
   following steps (these will be automated in future) NOTE: example for WCE5.0:

    1. Create the following folder WINCE60 PUBLIC root folder:
       C:\WINCE50\PUBLIC\COMMON\OAK\DRIVERS\DISPLAY\INTELIGD
    2. Copy ddi_igd.dll, ddi_igd.rel and video.reg into that folder
    3. Install igd_Wince50.cec file into Platform Builder by:
       - From the File menu, choose Manage Catalog Features.
       - Choose Import.
       - In the Import Catalog Features dialog box,
         choose the .cec file, and then choose Open.
       - From the View menu, choose Catalog
         to display the Catalog.


   WCE: Determining graphics memory size
   =========================================

   This version supports the static as well as dynamic location
   of graphics memory. For static video memory, the two registry
   settings ReservedMemoryBase and ReservedMemorySize will
```

```
denote where and how big this area will be. This portion
of memory will be taken care off by the built in
GART driver. The size should include the stolen memory
(BIOS setting, if applicable).

If either of these registry settings are zero or not
defined in the registry, then dynamic memory allocation will
be used for video memory management. The usage will be
dependant on system resources. This is in line with
Intel's Dynamic Video Memory Technology.

For example of static video memory, if the stolen memory
is 8 MB and you want a total of extra 56 MB of graphics
memory for a total of 64 Mbytes of graphics memory you
would want to have these settings: (On a 128MB machine)

[HKEY_LOCAL_MACHINE\Drivers\Display\Intel]
   "ReservedMemoryBase"=dword:4000000
   "ReservedMemorySize"=dword:4000000

Which means that the managed graphics memory pool will begin
at physical address 0x4000000 (64MB) and is 64MB of size.

Always remember to include the amount of stolen memory in
this number!

The lower 64MB in the above machine will be used for the NK.bin
and RAM, so therefore you must change your config.bib accordingly

The configuration for the given machine is (debug build):

    NK        80220000  01650000  RAMIMAGE
    RAM       81700000  02790000  RAM

As you see, the NK.BIN image takes 22.3125 MB and the rest of
41.6875 MB is RAM. The release build should have much smaller
NK area. For better control, users may turn AUTOSIZE=OFF to
calculate exactly where the image and RAM starts and ends followed
by the location of the video memory pool with no possibility of
runtime deviation.

These two settings, together with the registry settings mentioned
above will decide what your machines memory layout will look like.

The example config.bib and the memory configuration settings
in the video.reg are validated settings that can be used for static
video memory of 16MB in size starting from the 48MB address, thus
including the stolen memory.

WCE: Framebuffer and video surface size
===========================================
```

```
2 more optional registry settings are available to
limit the Framebuffer size of the display driver and
the total size of off-screen video surfaces.

The MaxFbSize registry entry will control the maximum
size of the framebuffer only. Actual usage will depend
on the mode being used.

The PageReqLimit registry entry will control the total
size in pages (4K) of all video surfaces, buffers allocated
for any use. Both of these registry configurations apply
to both the static as well as dynamic video memory management
explained in the previous section. The default below indicates
that a maximum of 2MB's are used for the framebuffer and
a maximum of 16MB's are permitted for all off-screen video
surface allocations.

   "MaxFbSize"=dword:200000
   "PageReqLimit"=dword:1000

In the case of Windows CE, because the OS does not allow
for dynamically setting the framebuffer size, the MaxFbSize
can be changed to match the mode setting being used in order
to minimize on video memory wastage. The following are
different suggested values for MaxFbSize for different display
modes. These values have not been validated. Note that 640X480
is calculated as 640X512 and 800X600 is calculated as 800X768
for stride alignment purposes.

 640X512X16 = A0000
 640X512X24 = F0000
 640X512X32 = 140000
 800X768X16 = 12C000
 800X768X24 = 1C2000
 800X768X32 = 258000
 1024X768X16 = 180000
 1024X768X24 = 240000
 1024X768X32 = 300000


WCE: Video surface allocation rule
====================================

Another two optional registries entries determine
a minimum width and height dimension that would enable
successful video surface allocations.

In Windows CE GDI, video surface allocations can happen
with a REQUIRE_VIDEO_MEMORY or a PREFER_VIDEO_MEMORY
flag. The following options will force surface
allocations with the PREFER_VIDEO_MEMORY flag to
```

```
be allocated in system memory if the width and height
are lower then what is stated in the following entries.

    "MinVidSurfX"=dword:10
    "MinVidSurfY"=dword:10
In this example, surfaces allocated with the
PREFER_VIDEO_MEMORY where the width and height
are both less than 16 pixels are forced to be
in system memory.


WCE: Escape API
===================
Some escape IOCTL code are exposed to user application space in order to control
non-mode-dependent port device attributes, such as Brightness, Contrast, etc,
raw I2C read/write, and port enable/disable. Please see the separate
 "IEGD API" document for details on the functions and their use.

In addition to the above, two Microsoft reserved Windows escape codes are also
supported in this release. They are SETPOWERMANAGEMENT and
GETPOWERMANAGEMENT.
However, in the implementation of GETPOWERMANAGEMENT, all power state requests
apart from VideoPowerOn will only equate to powering down of the display ports.



End WCE 5.0/6.0
***************



Software Errata
===============

This release also contains fixes for errata. For details on the errata,
including status information, refer to the Errata document usually located
at the following Web address:

 www.intel.com/design/intarch/specupdt/309380.htm

and also in the download section of the Intel Premier Support (Quad)
system.

Software Licenses
=================

The Intel Embedded Graphics Drivers are subject to the terms of the
license agreement located in the license file in this distribution package.
Please read, understand, and agree to the terms before using the IEGD drivers
or the IEGD Configuration Editor.
```

```
Legal / Disclaimers
===================

INFORMATION IN THIS DOCUMENT IS PROVIDED IN CONNECTION WITH INTEL(R) PRODUCTS.
NO LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE, TO ANY INTELLECTUAL
PROPERTY RIGHTS IS GRANTED BY THIS DOCUMENT. EXCEPT AS PROVIDED IN INTEL'S TERMS
AND CONDITIONS OF SALE FOR SUCH PRODUCTS, INTEL ASSUMES NO LIABILITY WHATSOEVER,
AND INTEL DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY, RELATING TO SALE AND/OR
USE
OF INTEL PRODUCTS INCLUDING LIABILITY OR WARRANTIES RELATING TO FITNESS FOR
A
PARTICULAR PURPOSE, MERCHANTABILITY, OR INFRINGEMENT OF ANY PATENT, COPYRIGHT
OR
OTHER INTELLECTUAL PROPERTY RIGHT. Intel products are not intended for use in
medical, life saving, life sustaining, critical control or safety systems, or
in
nuclear facility applications.

Intel may make changes to specifications and product descriptions at any time,
without notice.

(C) Intel Corporation 2008

* Other names and brands may be claimed as the property of others.

==========================================================================
======
```

第 8 章　WINCE 程序开发入门

Microsoft Windows CE 是紧凑的、高效的和可升级的操作系统，它被广泛地应用在各种嵌入式式的产品中，以及从手持电脑到专门的工业控制器和消费用电子产品中。Windows CE 已经通过了其自身的能力证明：它能够满足 32 位嵌入式式程序开发的需求。同样重要的是：Windows CE 使嵌入式式系统的设计者得以充分利用 Microsoft 的 32 位基于 Windows 的开发工具的全部函数。

嵌入式的应用软件选择 Windows CE 的首要原因之一是 Microsoft Win32 应用编程接口（API）的广泛应用。从运行在 Microsoft WindowsNT 操作系统下的高端服务器到最小的台式机和内嵌式的应用中，Win32 API 几乎是所有面向 Windows 开发 32 位应用软件的内核。

8.1　WINCE 下 Win32 应用程序介绍

对于全世界成千上万的程序开发者来说，Win32 程序模式很常见。Win32 是台式机或企业广泛应用的强大目标，而且随着 Windows CE 的涌现，它也成了嵌入式式系统理想的程序界面形式。Win32 为 Windows CE 操作系统提供了一致的，文档完备的，并且函数强大的程序界面。

用 Win32 在 Windows CE 平台开发的程序同其他的 Win32 程序差不多；这意味着程序开发者在开发新的面向 Windows CE 操作系统的应用程序的时候，能够应用这些大量的 Win32 的程序资源、第三代工具和外部的专门的技术。具有 Win32 编程经验的程序员会发现创建新的（维护已有的）面向 Windows CE 设备（如手持电脑）的应用程序，远比创建类似的面向特殊的平台或其他应用不广的嵌入式的操作系统的应用程序简单。

嵌入式系统的设计者们很快意识到 Windows CE 的强大，并且以惊人的速度使这个操作系统与它们的新产品一体化。可是，许多嵌入式式软件的开发者对 Windows 的事件驱动程序的一般技术不熟悉。对于这些软件开发者来说，在他们开始第一次的基于 Windows CE 的项目之前，了解一些 Win32 的基础知识是有好处的。

1. 什么是 Win32?

“Win32”表示对于所有的 Microsoft 32 位平台的一种普通的应用编程接口（API）。这些平台通常指：Windows 95、Windows 98、Windows NT、Windows2000、WindwsXP、Windows CE。

我们主要关注 Windows CE 操作系统，可是需要弄清楚的是大多数 Win32 的应用编程接口对上面所有三种平台都适用。这种通用的应用编程接口的优位有很多：容易登录到应用程序中，有大量的已有程序的知识，范例和第三代软件的资源。

2. 平台的差别

Win32 的应用编程接口定义了作为一个程序员有用的 Windows 平台。Win32 应用编程接口的目的是提供一个常用的界面的设置，可以是虚拟的。不同的平台由于不同的特性及硬件的约束，其应用编程接口也不同。Win32 平台家族的一些成员支持全部 Win32 的应用编程接口（下简称 API），而其他的只支持一部分的 API。为紧凑的、嵌入式式的应用软件和小型设备而设计的 Windows CE，Win32 的 API 最受限制。尽管如此，Windows CE 的 API 也是足够完善的，能够处理实际的、大量的、高级的嵌入式式应用程序。

3. Win32 与 Microsoft 基本类库（MFC）

用 Win32 界面设计并不是创建 32 位基于 Windows 的应用程序的唯一途径。另一种重要的途径是可在 Win32 与 Microsoft Visual C++开发环境使用的 Microsoft 基本类库（MFC）。MFC 为许多（不是所有）的 Win32 的 API 进行了高度的封装。通常，MFC 提供了代表重要的 Windows 的用户界面对象的类，像窗口、对话框、画刷、画笔和字体等。MFC 也为没有任何用户界面要求的嵌入式式应用软件提供了相应的类。MFC 类的成员函数调用 Win32API 的函数，可以使复杂的应用程序的设计巧妙简化。

作为 Win32 的程序员，可以自由地选取使用 C 或 C++和 Win32 API，或者用 C++与 MFC。

4. Win32 程序模型

Win32 是超越所有 32 位 Windows 平台的常用和一致的（尽管并不全部相同）API。为了对 Win32 API 有一个透彻的了解，以便高效地使用它的函数，了解一些底层操作系统的基础十分重要。

8.2 WINCE 下 32 位 API 的调用

1. Windows CE 的内核和 Win32 API

那些对 Windows CE 不了解的人可能会有一个错觉，认为它仅仅是现有的操作系统（如 Windows 95）降级版。但事实上 Windows CE 是以小型的、高度用户化的面向嵌入式式应用程序的操作系统开发起来的。在 Windows CE 排除（或代替）了一些基于 Windows CE 的应用软件所不需要的操作系统特性的同时，它的内核也具有大量的其他 Microsoft 32 位的操作系统最精华的东西。例如在 Windows NT 下，所有在 Windows CE 下运行的应用软件都运行在有优先权的多任务处理环境下，在被全保护的内存空间里。还有，像 Windows NT 一样，Windows CE 支持本地统一的字符编码标准码字符串，使它更适于国际推广。可是，不像其他的 32 位 Windows 平台，Windows CE 是十分紧凑和用户化的，仅仅占用小于 200 KB 的内存。

Windows CE 的 Win32 API 比其他的 32 位的 Windows 操作系统的 Win32 API 要小；它只包括大约相当于 Windows NT 的半数的 API。但是 Windows CE 的 Win32 API 也有其他系统所没有的特性。例如：通知 API，它能够操作系统的层次，而不是在运行的应用程序的层次上处理通知事件（如时钟）。触摸屏的 API 和对 Windows CE 数据库内置的支持是其他操作系统所没有的。另一个关于 Windows CE 鲜为人知的方面是它高度的模块化；嵌入式系统开发者（用 Microsoft Windows CE 内含的面向 Visual C++的软件包）能够创建一个对于它们独特的硬件平台和应用软件用户化了的 Windows CE 的版本。

Windows CE 操作系统为设计者提供了全新的设计环境。Windows CE 的开发者几乎不需要支持原有的应用程序或设备，所以操作系统在设计时可以考虑到应用一些最新的思想和应用程序，并且应用最先进的嵌入式式的 32 位微处理器产品作为它的硬件目标平台。

这对于一个使用 Windows CE 平台的 Win32 API 的用户来说意味着什么呢？这意味着对于现代的、32 位的嵌入式式系统，它是一种更简单的 API，更加优化的目标操作系统。下面，我们将研究一些重要的 Win32 API 和 Windows CE 操作系统内部的内容。

2. 程序和线程

了解 Win32 API 和 Windows CE 操作系统的底层内容的第一步就是了解多任务和多线程应用程序是如何组织的。Win32 术语中，程序被定义成一个正在运行中的程序实例。像其他

的 32 位 Windows 平台一样，Windows CE 是一个多任务的操作系统，在一个运行中的程序中，它支持执行多个线程索。

对于嵌入式式的应用程序，Windows CE 的多线程执行能力是它函数重要的体现。这样就使 Win32 嵌入式式程序开发者优先考虑 Win32 的线程索的创立与同步。

Win32 线程索的处理同其他常用的嵌入式式的操作系统是有区别的。不像 UNIX 或其派生的系统，32 位 Windows 平台从一开始设计就支持多线程应用程序。线程索管理（时序安排，同步和资源管理）由内核来完成，程序开发者利用函数装入内核（通过 Win32 API 访问）来建立和管理他们应用程序中的线程索。

例如，如果一个嵌入式式应用程序必须监视多个输入设备并且在监视到一个或多个设备上不同步发生的事件时要作出恰当的反映。更进一步，如果这样一个程序也需要更新一些共享的资源（如全局数据结构，磁盘上的文件，或其他设备）来作为与相关设备事件的反映。像这样的一个程序需要一个可靠的线程索管理系统。这恰恰是 Win32 API 能够提供的函数：多个线程索能够快捷并容易地用 Win32 API 线程索创建界面建立；同步线程索（多个线程索同时访问一个数据）可以通过不同的方式完成，包括关键的段，有名称和没有名称事件，以及互斥的目标。

Windows CE 被设计成在执行这些同步时占用最少的程序资源。这对那些函数不强大的开发者来说十分重要；因为内核来负责线程索的管理，不需要使用另外的处理器来循环检测程序或线程索完成，以及执行其他的无用的应用程序层次上的线程索管理。内核已构建好如何管理若干个线程索并且使程序高效地进行的程序。

对于包括若干个程序的应用程序，Win32 向用于线程索、程序管理和同步提供了一套完备的处理方法。这些线程索管理特性非常适于嵌入式式应用程序软件，并且对 Windows CE 开发者是容易得到的。

3. 消息

在 32 位 Windows 平台上运行的程序更专业化，程序的线程依赖于消息来初始化程序，控制系统资源并与操作系统和用户通信。Windows 消息有各种各样的来源，包括操作系统，用户活动诸如键盘输入、鼠标、触到屏幕，以及其他运行的程序或线程。

当消息被送到线程时，这条消息被放置在消息队列中等待最后处理。每一条线程拥有完全不独立于其他线程所拥有的消息队列的消息队列。线程一般有不断运行的消息循环，恢复和处理消息。当队列没有消息，并且线程不从事其他任何活动时，系统挂起线程，以节省中央控制器资源。

消息也能用于控制目的，初始化应用程序中各种类型的程序，并且它们能利用消息参数传递数据。例如，线程可能收到触摸屏被激活的消息，消息参数可以表明 X 和 Y 为用户行动的坐标。在另一种类型的消息中，参数可以包括指针或者指向数据结构、窗口或其他对象的句柄。

4. 中断处理

作为一个嵌入式的软件开发者，用户可能最关心 Windows CE 消息的处理规则是如何影响自己的外部系统接口的时序的。Windows CE 通过细心设计和准确的衡量以保证其中断时序以及其他相关的特位与嵌入式式的系统设计是适用的。

嵌入式应用程序经常有时间临界的设备接口需要，需要发现并且在一段最小的规定时间之内对设备和系统事件作出反应。为了支持这样的应用程序，Windows CE 包括高度优化中断传送，优先级和服务系统。

在 Windows CE 内核中，中断处理分成两个明显的部分：中断服务程序（ISR）以及中断服务线程（IST）。这个系统的目的是使 ISR 尽可能小和快。在硬件的层次上，每一中断要求（IRQ）线程与一特定的软件 ISR 联系。当被触发时，给定的 ISR 除了通知内核 IST 的位置外，还做少量的工作。一旦 IST 被初始化（尽管没必要完成），系统便准备好接受下一中断并且处理下一中断。

每一个中断有一个优先级与它们相联系。Windows CE 为确定的线程时序，利用基于优先级的时间片段算法。与每一 ISR 被联系的 IST 是正常的线程，因此为 IST 设置优先级以满足应用程序的时序需要是应用程序软件开发者的责任。

这种将 ISR 和 IST 在中断程序中分开处理的最终结果是，典型的中断等待时间被大大地减少了，在中断程序中发生不可接受的延迟的可能性也大大减少。

5. 内存管理

Win32 API 向开发者提供了一套完备的和一致的接口。当开发绝大部分应用程序时，软件开发者不需要考虑特定内存结构。然而，对于许多嵌入式应用程序，特别是那些有严格的内存资源约束或者临界时序的要求的，对内存被管理的方法有好的理解是重要的。

Windows 内存的一般结构对于不同的 32 位 Windows 平台是不同的，并且特殊的细节结构在同一 32 位 Windows 操作系统下不同的处理机之间也不同（例如，Windows NT 的内存结构在 X86 平台上与在 DEC Alpha 平台上的用法十分不同）。对于这段的讨论，我们将专门集中在 Windows CE 操作系统的部分中进行。

6. Windows CE 的存储结构

像其他的 32 位 Windows 平台一样，Windows CE 操作系统也有虚拟内存的特性。内存总在某一时间被分配给应用程序一页，页的大小由系统设计者决定（并在操作系统为目标硬件平台创建时被指定）。例如在手持电脑上内存页大小是典型的 1 KB 或者 4 KB。

在初始化期间（导入），Windows CE 创造一个独立的被所有程序共享的 4 GB 虚拟地址空间。当程序引用一个虚拟的地址时，它被内核记录在物理的内存上。这使得应用程序软件开发者不必去考虑目标系统内存的物理布局。虽然所有程序共享单一地址空间，应用程序仍然可避免相互误用。Windows CE 通过改变每页的保护来保护程序内存，而不是分配给每一程序不同地址空间。作为应用程序开发者，可能不会太在乎目标系统内存的物理结构。内存可以全部是随机存取内存，或者它可能包括闪存卡或者硬盘驱动器。Windows CE 操作系统为用户管理内存资源，同时 Win32 API 向用户提供必要的分配、使用和释放内存的接口。然而，作为一个嵌入式系统的设计者，用户将需要细心考虑将在新的硬件平台上执行的应用程序的内存需要，并且全面考虑成本、速度和可靠性，平衡各种目标的冲突。如果用户为使用 Windows CE 开发一个新的硬件平台，Windows CE 的面向 Visual C++的嵌入式软件包包括的资源可以帮助你做出这些决策，从而构成操作系统。

无论用户系统内存的配置如何，ROM（只读内存）将占用十分重要的地位。不同于其他的 32 位 Windows 操作系统，Windows CE 操作系统的代码在只读内存中，并且在只读内存中原地执行。依据你的产品需要，也能选择在只读内存中放置应用程序代码。例如，Pocket Word，Pocket Excel 和其他应用程序，包括在手持电脑只读内存中被提供的。

存储在 ROM 中的程序组在 Windows CE 下当地执行，所以嵌入式系统的设计者能够只占用很少的 RAM 用于堆栈存储的需要。相应地，你的嵌入式应用程序可以利用 RAM 既作

为程序的内存又可做临时存储空间。

为了进一步增加应用程序软件的性能， Windows CE 采用按需求将内存分页；操作系统仅仅需要解压缩并且装入基于 RAM 的一小部分程序准备执行。ROM 和基于 RAM 的程序的灵活性与速度意味着基于 Windows CE 的设备能够被构造成各种内存结构形式。

7. 手持电脑的内存结构

典型的 Windows CE 的硬件平台的内存结构是与基于 Windows 系统的台式电脑的内存结构不同的。为了知道内存通常如何在 Windows 中被处理，考查基于 Windows CE 的最普通的代表性的设备——手持电脑，是很有用的。

在手持电脑中，RAM 被分割成两个主要的部分：存储内存和程序内存。向两部分分配的 RAM 的量能被手持电脑用户修改（在限制范围内）。

在手持电脑中的存储内存类似于台式电脑的硬盘 RAM。它被用来存储数据和非系统应用程序。它的三段中每一段被不同的一套 Win32 API 访问函数。

Windows CE 系统寄存器类似于 Windows NT 和 Windows 95 的操作系统的寄存器。用户可以利用 Win32 寄存器函数来操纵寄存器中的键和数值。

被用户安装的应用程序和数据在一般文件存储段中。Windows CE 文件系统 API 是标准 Win32 文件系统的子集函数。

对于数据库应用程序，由 Windows CE 数据库 API 来存储被管理存储。这 API 对 Windows CE 是唯一的，并且在其他的 Win32 平台中没有。

程序内存被用于系统和非系统程序的堆栈存储。非系统应用程序从存储内存（或者或 PC 卡）中被取得，非压缩的并且被装入要执行的程序内存中。

8. 意外情况处理

意外情况处理是强大的编程技术，相应的 Win32 API 起函数能容易地发现未预料到的错误状况，并且使之恢复。结构化的意外情况处理，允许危险的段的代码可能由于硬件资源的问题、设备的冲突和微小的编码错误而导致失败，以使这部分程序与其余的应用程序分开。这保护了应用程序，使之免于过早地终止或者产生敏感的系统问题。

结构化的意外情况处理包括定义一系列声明作为保护，并且为第一套的声明定义了另一个套声明作为意外情况句柄。意外情况句柄定义了一个或多个声明来保障系统的运行，而不管保护声明现有的状态。

在大多数 32 位 Windows 平台上应用 Win32 API 的程序员在运用意外情况句柄时通常有两种选择，用 C 或 C++编写应用程序，并且利用 Win32 提供的处理意外情况的宏，或者利用 C++ 编写应用程序，并且使用 C++ 语言定义的意外情况处理函数。

对于这种程序的编写，Windows CE 的开发者因无法访问 C++的（面向 Windows CE 的 Visual C++目前还不支持）意外情况处理，所以必须使用 Win32 API 的意外情况处理宏。

为了应用 Win32 意外情况处理，可以使用一套在 Win32 API 中被定义的宏。下面一段代码显示其基本概念：

```
_try {
// The statements in here have a possibility of failure
// and so are guarded.
}
__finally {
// This is the exception handler. This code will execute
// after the guarded statements, no matter what happened
// in the guarded block of code above.
```

```
}
// This code will execute normally if the program flow allows
// it (no goto, exit, etc.)
```

__try 以及__finally 宏产生了使用意外情况句柄的所必要的底层代码。

意外情况的处理对诸如在嵌入式的应用程序中的那些普通的多线程序是有用的。Win32 结构化意外情况处理宏是一种容易并且强大的，保护应用程序使之免受未预料到的失败的方法。

9. 设备处理

有无数硬件设备（外围设备）与应用 Windows（Windows NT 以及 Windows 95）平台的台式机是兼容的，并且每一年都有更多的东西在市场上涌现。而 Windows CE 的平台，通常不支持台式计算机支持的设备的很多品种的外围硬件。然而，为一嵌入式的系统创造可靠的设备接口在嵌入式的程序设计的过程中，是比较富有挑战性的部分。这是因为典型的嵌入式系统接口的时序与其他可操作性的需要远比台式计算机计算系统和应用程序的更难、更复杂。

幸运的是，Win32 API 提供了一套丰富的设备接口方法，使得设备接口程序写起来更容易并适合特定嵌入式的系统的需要。

Win32 API 是如何帮助 Win32 API 在你的硬件平台为你提供一套一致的基于流的接口。为了使用设备，用户首先利用适合设备类型的函数打开它。对于大多数设备，利用的函数是下列例子中的 CreateFile 函数：

```
HANDLE hPort = CreateFile("COM1"); // Open the serial port
```

CreateFile 函数打开规定的设备（串口）并且返回用于以后在该种设备上的操作（例如读和写）的句柄。各种各样的函数（包括 ReadFile、WriteFile、LockFile 和其他）接受这个句柄为参数，并且允许读写数据，检查设备状态，并且将从其他程序的存取被锁住的设备或者文件列入清单。文件输入输出操作被处理成与其他设备类型利用同样的 API 函数，并且在文件之内包括随机访问的函数。被若干程序或线程同时访问的设备和文件可以分区域地利用 LockFile 函数锁定。

在用户的应用程序已完成设备或者文件之后，它将调用 CloseFile 函数关闭设备，并且进行必要的清除设备的工作。

10. 同步和异步的设备的处理

嵌入式系统经常有关键的设备有时序需要。对于这个理由，对底层的操作系统的软件接口必须能够在软件层次上管理同时（或者几乎同时）的系统中不同类型的设备的事件。Win32 API 支持对设备的同步和异步的访问，并且用复杂的设备接口设计。

同步的接口是那些在软件需要从设备得到动作的要求，然后等候结果。在同步的设备接口中，最常用的是前面已经提到的 ReadFile 以及 WriteFile 函数。当在同步 I/O 中使用时，不论与磁盘上的文件、并口或是串口、一个通道或其他类型的设备接口时，都是公用的并且是兼容的。

异步的接口是那些设备要求应用程序为之服务的接口。一个异步的设备的好的例子是键盘。适当和适时地处理异步事件，对于许多嵌入式应用程序是至关紧要的。幸运的是，为 Windows CE 编写的设备驱动程序能支持同时多线程访问驱动器。这大大地简化异步输入设备的处理。

所访问的给定的设备的方法，取决于那个设备的特性和开发的特定的应用程序的要求。如果在基于 Windows CE 的硬件平台上创建一个全新的设备（和设备驱动程序），既可从硬件设备和驱动器的层次，又可从应用程序的层次有许多选择。

第 9 章　用 Platform Builder 开发程序

9.1　Platform Builder 开发 DLL 的流程

在 Windows CE 中，所有的驱动程序都以 dll 形式存在。dll 文件可以用 EVC 来开发，也可以使用 PB 来开发，使用 PB 开发驱动程序，可以与 NK 同时进行编译，要比 EVC 来的方便一点。

使用 PB 来开发，首先应该在工作平台下面建立一个目录，用来存放源文件，同时要修改 dir 文件，使得编译的时候能够进到源文件所在的目录。

这里主要解释一下使用 PB 编译过程中需要增加的文件。

第一个文件是 sources 文件，这里给出了一个 sources 文件的例子：

TARGETNAME=led

使用 TARGETNAME 来指示生成目标的文件名（不包含扩展名，扩展名 PB 会自动加上）

RELEASETYPE=PLATFORM

RELEASETYPE 指示该文件将要生成的类型，一共有 5 个取值，分别如下。

SDK：使用该类型将使得生成的目标文件被存储到%_PUBLICROOT%\Oak 目录，而 lib 文件被放置到%_PUBLICROOT%\Sdk 目录。

DDK：使用该类型将使得生成的目标文件被存储到%_PUBLICROOT%\Oak 目录，而 lib 文件被放置到%_PUBLICROOT%\Ddk。

PLATFORM：使用该类型将使得生成的文件受平台控制。

LOCAL：该类型使得生成的文件全部放置到当前路径。

CUSTOM：该类型使得生成的文件放置到 TARGETPATH 制定的位置（也就是说必须要有 TARGETPATH 参数设置）。

TARGETTYPE=DYNLINK

生成的目标类型，LIBRARY 表示是一个 lib 库，DYNLINK 则表示是 dll，而 PROGRAM 则是一个 exe 文件。

TARGETLIBS=\
　$（_COMMONSDKROOT）\lib\$（_CPUINDPATH）\coredll.lib

TARGETLIBS 指示连接需要的库的名字

SOURCELIBS=mm.lib

SOURCELIBS 指示将于某一个 lib 一起连接。上面一个 lib 是需要什么就取什么，而这个 lib 则是连接所有的。

DEFFILE=led.def

DLL 文件的 def 文件名

INCLUDES=..\..\inc

指定 include 的路径

SOURCES= \

led.c

指定参与该规则的所有的源文件

第 2 个则是 makefile 文件，该文件只需要!INCLUDE $(_MAKEENVROOT)\makefile.def 就可以了，建议不要修改（我也不知道如何修改）

第 3 个文件是 def 文件，下面是一个该文件的实例

LIBRARY　　　led

生成的库（.lib）文件名（PB 自动生成扩展名）

EXPORTS

　mm_init

EXPORTS 指定了需要导出的函数名称

建立好这些文件之后，选择 PB 的 build 菜单的 open build releasee directory，到达所在的目录，执行 build　-c，如果没有错误，就按照我们的要求生成文件了。

通常，使用 build -c 只能把文件放到我们指定的路径，如果要把文件放到 NK 目录中，需要重新 build 或者手工复制到那个目录。

9.2　用 Platform Builder 开发标准流驱动框架

步骤 1 如图 9-1 所示，打开 WINCE6 的工程，选择“Add New Subproject…”，新建一个工程。

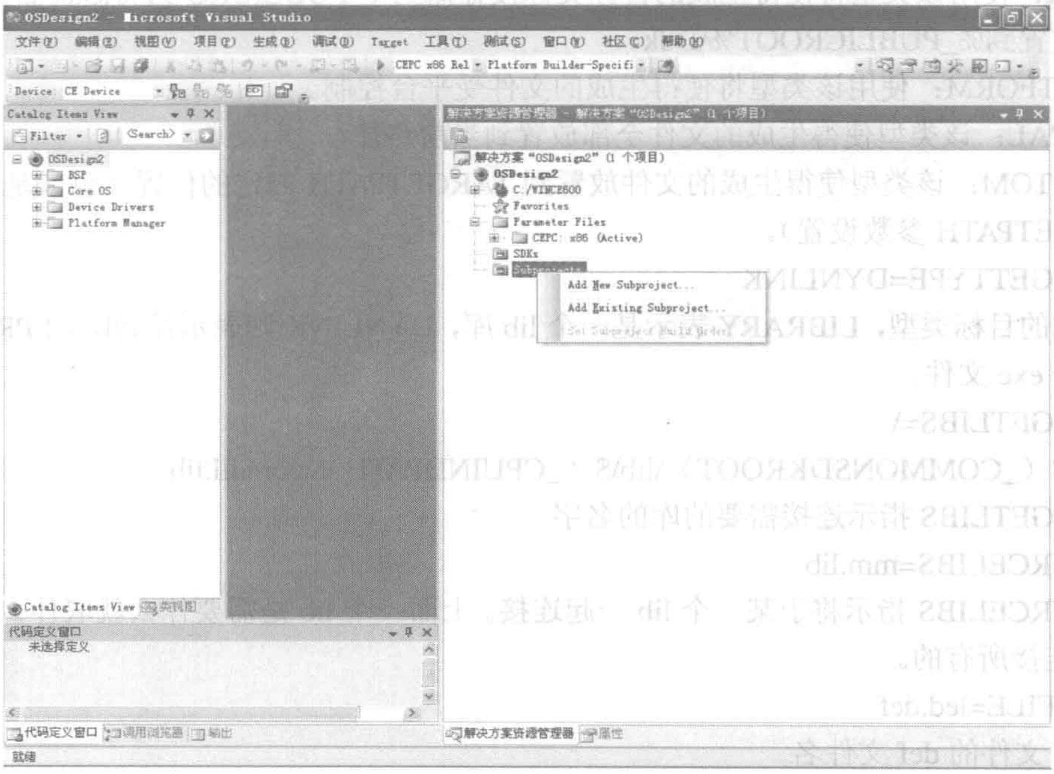


图 9-1

步骤 2　如图 9-2 所示，选择动态链接库工程，并将名称设置为“StreamDrv”。

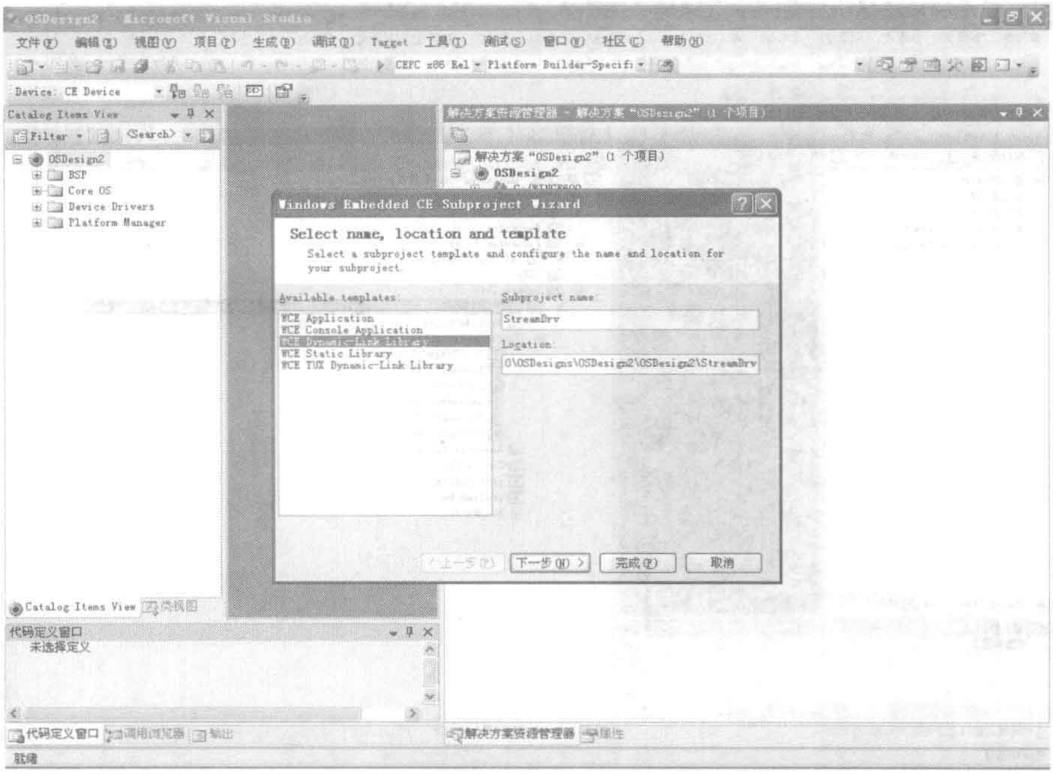


图 9-2

步骤 3　如图 9-3 所示，选择新建一个空项目。

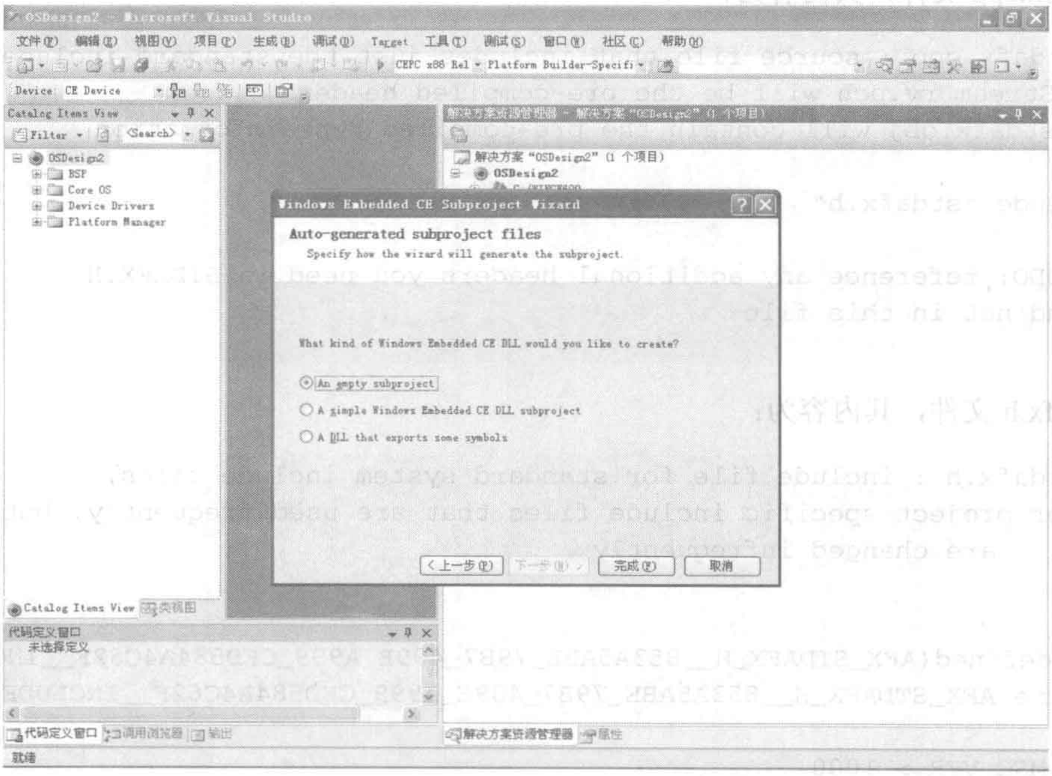


图 9-3

步骤 4 如图 9-4 所示，手工添加下列文件。

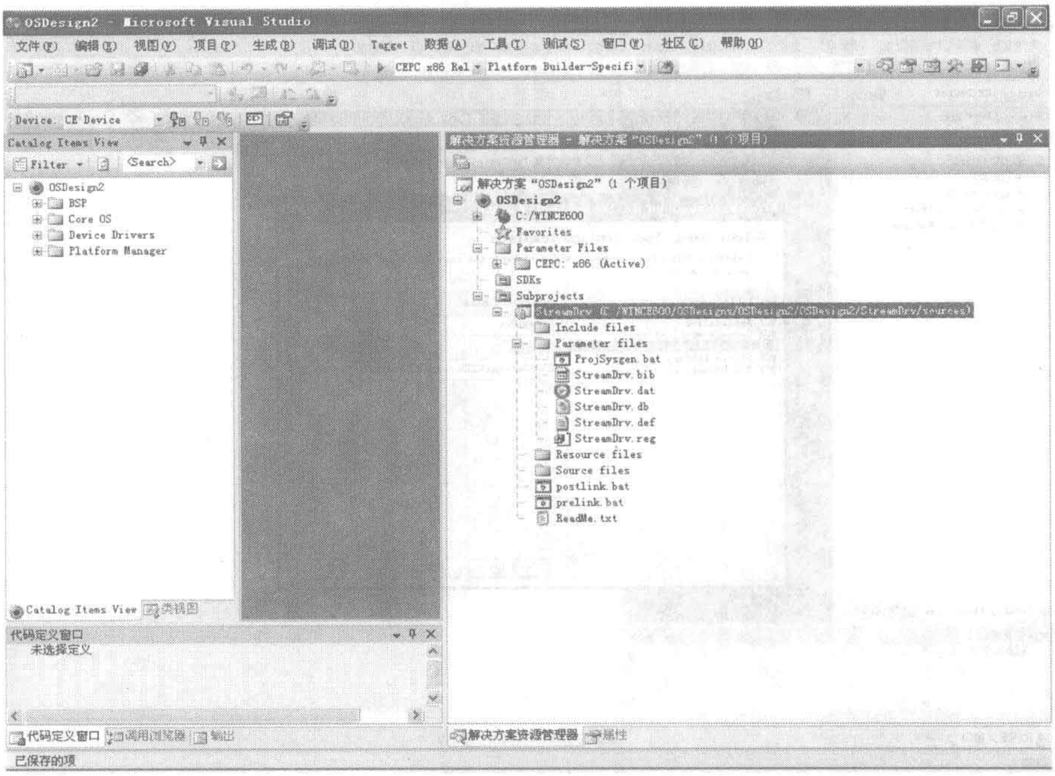


图 9-4

在空项目中手工加入以下文件。

StdAfx.cpp 文件，其内容为：

```
// stdafx.cpp : source file that includes just the standard includes
//   StreamDrv.pch will be the pre-compiled header
//   stdafx.obj will contain the pre-compiled type information

#include "stdafx.h"

// TODO: reference any additional headers you need in STDAFX.H
// and not in this file
```

StdAfx.h 文件，其内容为：

```
// stdafx.h : include file for standard system include files,
//  or project specific include files that are used frequently, but
//      are changed infrequently
//

#if !defined(AFX_STDAFX_H__853A5ABE_79B7_409E_A999_CED584A4C62F__INCLUDED_)
#define AFX_STDAFX_H__853A5ABE_79B7_409E_A999_CED584A4C62F__INCLUDED_

#if _MSC_VER > 1000
#pragma once
#endif // _MSC_VER > 1000
```

```
// Insert your headers here
#define Win32_LEAN_AND_MEAN     // Exclude rarely-used stuff from Windows
headers

#include <windows.h>

// TODO: reference additional headers your program requires here

//{{AFX_INSERT_LOCATION}}
// Microsoft Visual C++ will insert additional declarations immediately before
the previous line.

#endif
// !defined(AFX_STDAFX_H__853A5ABE_79B7_409E_A999_CED584A4C62F__INCLUDED_)
```

StreamDrv.cpp 文件，其内容为：

```
// This is a Sample Driver
#include "stdafx.h"
DWORD DEM_Init(LPCTSTR pContext, LPCVOID lpvBusContext);
BOOL DEM_Deinit( DWORD hDeviceContext );
DWORD DEM_Open( DWORD hDeviceContext, DWORD AccessCode, DWORD ShareMode );
BOOL DEM_Close( DWORD hOpenContext );
BOOL DEM_IOControl( DWORD hOpenContext, DWORD dwCode, PBYTE pBufIn, DWORD
dwLenIn, PBYTE pBufOut, DWORD dwLenOut, PDWORD pdwActualOut );
void DEM_PowerUp( DWORD hDeviceContext );
void DEM_PowerDown( DWORD hDeviceContext );
DWORD DEM_Read( DWORD hOpenContext, LPVOID pBuffer, DWORD Count );
DWORD DEM_Write( DWORD hOpenContext, LPCVOID pBuffer, DWORD Count );
DWORD DEM_Seek( DWORD hOpenContext, long Amount, WORD Type );
BOOL CustomFunction(void);
BOOL CustomFunctionEx(LPTSTR lpBuffer);
void DBGOut(DWORD dwValue);

#define IOCTL_DRIVER_REVSTR 0x1000

HANDLE hMem;        // Storage for Inbound Strings
DWORD  dwCount;     // Size of current String
// ----------------------------------------------------

BOOL APIENTRY DllMain( HANDLE hModule, DWORD  ul_reason_for_call, LPVOID
lpReserved )
{
    switch(ul_reason_for_call) {
        case DLL_PROCESS_ATTACH:
            OutputDebugString(L"StreamDrv - DLL_PROCESS_ATTACH\n");
        break;
        case DLL_PROCESS_DETACH:
            OutputDebugString(L"StreamDrv - DLL_PROCESS_DETACH\n");
        break;
```

```
        case DLL_THREAD_ATTACH:
            OutputDebugString(L"StreamDrv - DLL_THREAD_ATTACH\n");
        break;
        case DLL_THREAD_DETACH:
            OutputDebugString(L"StreamDrv - DLL_THREAD_DETACH\n");
        break;
        default:
        break;
    }
return TRUE;
}
// Driver Init...
DWORD DEM_Init( LPCTSTR pContext, LPCVOID lpvBusContext)
{
  OutputDebugString(L"StreamDrv - DEM_Init - Context: ");
  OutputDebugString(pContext);
  OutputDebugString(L"\n");
  hMem=LocalAlloc(LPTR,0x2000);   // 0x1000 Unicode Characters

  OutputDebugString(L"DemoDriver - ~ DEM_Init\n");
  return 0x1234;
}

BOOL DEM_Deinit( DWORD hDeviceContext )
{
OutputDebugString(L"StreamDrv - DEM_Deinit\n");

OutputDebugString(L"StreamDrv - ~ DEM_Deinit\n");
return TRUE;
}

// Driver Open
DWORD DEM_Open( DWORD hDeviceContext, DWORD AccessCode, DWORD ShareMode )
{
  OutputDebugString(L"DemoDriver - DEM_Open\n");
  OutputDebugString(L"hDeviceContext - ");
  DBGOut(hDeviceContext);
  OutputDebugString(L"\n");
  OutputDebugString(L"DemoDriver - ~ DEM_Open\n");
return 0x5678;
}
BOOL DEM_Close( DWORD hOpenContext )
{
  OutputDebugString(L"StreamDrv - DEM_Close\n");
  OutputDebugString(L"hOpenContext - ");
  DBGOut(hOpenContext);
  OutputDebugString(L"\n");
  OutputDebugString(L"StreamDrv - ~ DEM_Close\n");
return TRUE;
}
BOOL DEM_IOControl( DWORD hOpenContext, DWORD dwCode, PBYTE pBufIn, DWORD
```

```
dwLenIn, PBYTE pBufOut, DWORD dwLenOut, PDWORD pdwActualOut )
  {
    OutputDebugString(L"StreamDrv - DEM_IOControl\n");
    OutputDebugString(L"hOpenContext - ");
    DBGOut(hOpenContext);
    OutputDebugString(L"\n");
    switch (dwCode) {
      case IOCTL_DRIVER_REVSTR:
      {
         OutputDebugString(L"DRIVER REVSTR IOCTL...\n");
         // reverse the string...
         HANDLE hTemp=LocalAlloc(LPTR,dwLenIn+sizeof(TCHAR));
         memset(hTemp,0x00,dwLenIn+1);
         TCHAR *tcOut=(TCHAR*)hTemp;
         TCHAR *tcIn=(TCHAR*)pBufIn;
         DWORD dwChars=dwLenIn/2;
         for (DWORD x=0;x < dwChars;x++) {
              tcOut[x]=tcIn[dwChars-x-1];
         }
         memcpy(pBufOut,hTemp,dwLenIn);
         LocalFree(hTemp);
         *pdwActualOut=dwLenIn;
      }
      break;
      default:
         OutputDebugString(L"Unknown IOCTL\n");
      break;
  }
  OutputDebugString(L"StreamDrv - ~ DEM_IOControl\n");
  return TRUE;
  }
  void DEM_PowerUp( DWORD hDeviceContext )
  {
    OutputDebugString(L"StreamDrv - DEM_PowerUp\n");
    OutputDebugString(L"hDeviceContext - ");
    DBGOut(hDeviceContext);
    OutputDebugString(L"\n");

    OutputDebugString(L"StreamDrv - ~ DEM_PowerUp\n");
  }

  void DEM_PowerDown( DWORD hDeviceContext )
  {
    OutputDebugString(L"StreamDrv - DEM_PowerDown\n");
    OutputDebugString(L"hDeviceContext - ");
    DBGOut(hDeviceContext);
    OutputDebugString(L"\n");

    OutputDebugString(L"StreamDrv - ~ DEM_PowerDown\n");
  }

  DWORD DEM_Read( DWORD hOpenContext, LPVOID pBuffer, DWORD Count )
```

```
{
  DWORD dwRetCount=0xffff;      // default to error
  OutputDebugString(L"StreamDrv - DEM_Read\n");
  OutputDebugString(L"hOpenContext - ");
  DBGOut(hOpenContext);
  OutputDebugString(L"\n");
  if (NULL != hMem) {
     dwRetCount=dwCount;
     memcpy(pBuffer,hMem,dwCount);
  }
  OutputDebugString(L"StreamDrv - ~ DEM_Read\n");

return dwRetCount;
}

DWORD DEM_Write( DWORD hOpenContext, LPCVOID pBuffer, DWORD Count )
{
  OutputDebugString(L"StreamDrv - DEM_Write\n");
  OutputDebugString(L"hOpenContext - ");
  DBGOut(hOpenContext);
  OutputDebugString(L"\n");

  if (NULL != hMem) {
     LocalFree(hMem);
  }

  hMem=LocalAlloc(LPTR,Count);
  memcpy(hMem,pBuffer,Count);
  dwCount=Count;

  OutputDebugString(L"StreamDrv - ~ DEM_Write\n");

return Count;
}

DWORD DEM_Seek( DWORD hOpenContext, long Amount, WORD Type )
{
  OutputDebugString(L"StreamDrv - DEM_Seek\n");
  OutputDebugString(L"hOpenContext - ");
  DBGOut(hOpenContext);
  OutputDebugString(L"\n");

  OutputDebugString(L"StreamDrv - ~ DEM_Seek\n");

return 0;
}
BOOL CustomFunction(void)
{
  OutputDebugString(L"StreamDrv - CustomFunction\n");
  OutputDebugString(L"StreamDrv - ~ CustomFunction\n");
return TRUE;
}
```

```
BOOL CustomFunctionEx(LPTSTR lpBuffer )
{
  OutputDebugString(L"StreamDrv - CustomFunctionEx\n");
  TCHAR buffer[16];
  _tcscpy(buffer, lpBuffer);
  OutputDebugString(buffer);
  OutputDebugString(_T("\n"));
  OutputDebugString(L"StreamDrv - ~ CustomFunctionEx\n");
return TRUE;
}

void DBGOut(DWORD dwValue)
{
  TCHAR tcTemp[10];
  wsprintf(tcTemp,L"%ld",dwValue);
  OutputDebugString(tcTemp);
}
```

修改以下文件。

StreamDrv.def 文件，其内容为：

```
LIBRARY DemoDriver

EXPORTS
   DEM_Init
   DEM_Deinit
   DEM_Open
   DEM_Close
   DEM_IOControl
   DEM_PowerUp
   DEM_PowerDown
   DEM_Read
   DEM_Write
   DEM_Seek
   CustomFunction
   CustomFunctionEx
```

StreamDrv.reg 文件，其内容为：

```
[HKEY_LOCAL_MACHINE\Drivers\BuiltIn\StreamDrv]
   "Dll" = "StreamDrv.Dll"
   "Prefix" = "DEM"
   "Order" = dword:0
   "FriendlyName" = "StreamDrv Driver"
   "Ioctl" = dword:0
```

步骤 5 如图 9-5 所示，添加完文件之后，单击右键，查看项目属性，根据需要正确设置编译选项（编译选项的设置同 VC 类似，这里不作说明）。

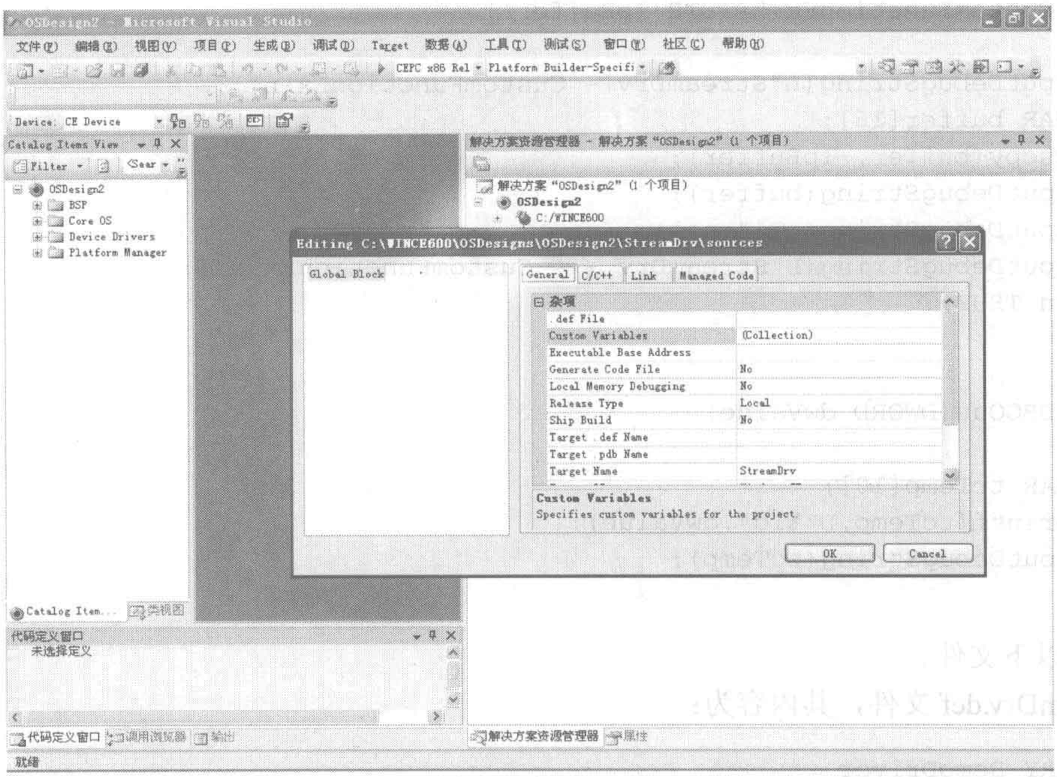


图 9-5

步骤 6 如图 9-6 所示，选中该工程，单击右键，在弹出的快捷菜单中选择“Build”或者“Rebuild”选项，即可编译当前工程，得到最终生成的结果 DLL。这样，我们就得到了一个标准流驱动的框架。

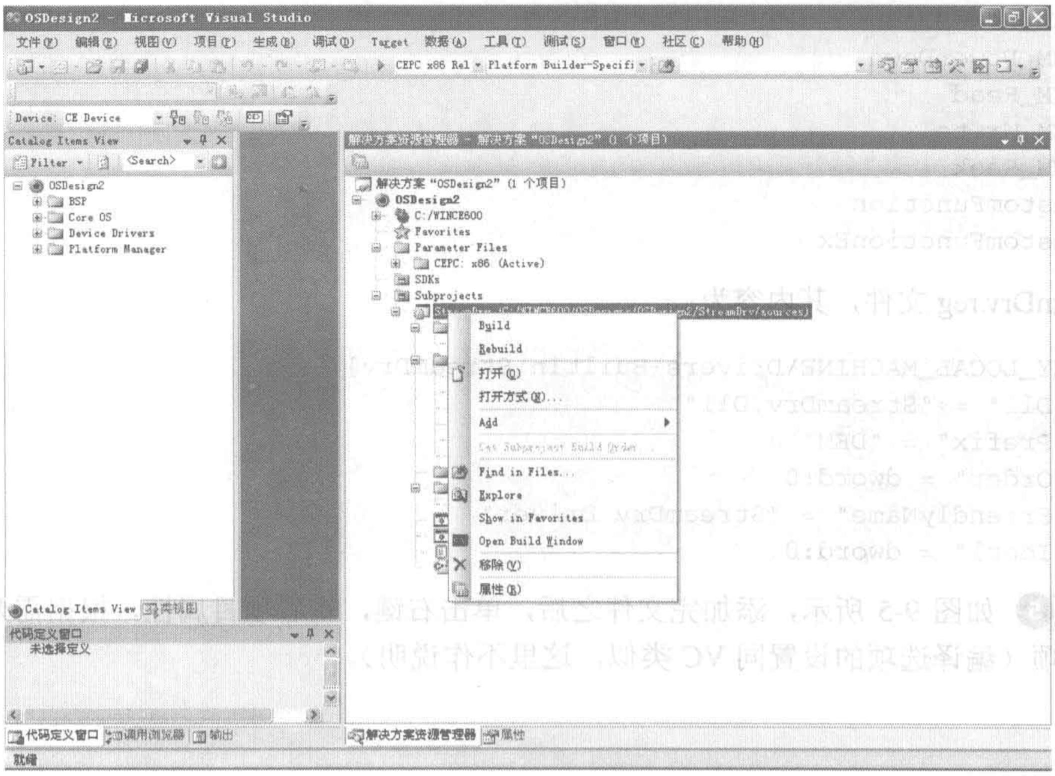


图 9-6

9.3 Platform Builder 调用流驱动程序示例

如图 9-7 所示，在 PLATFORM BUILDER 平台中根据向导新建一个应用程序，名称为 “TuxTest”（过程与新建立 DLL 几乎全部相同，因此不再截图）。

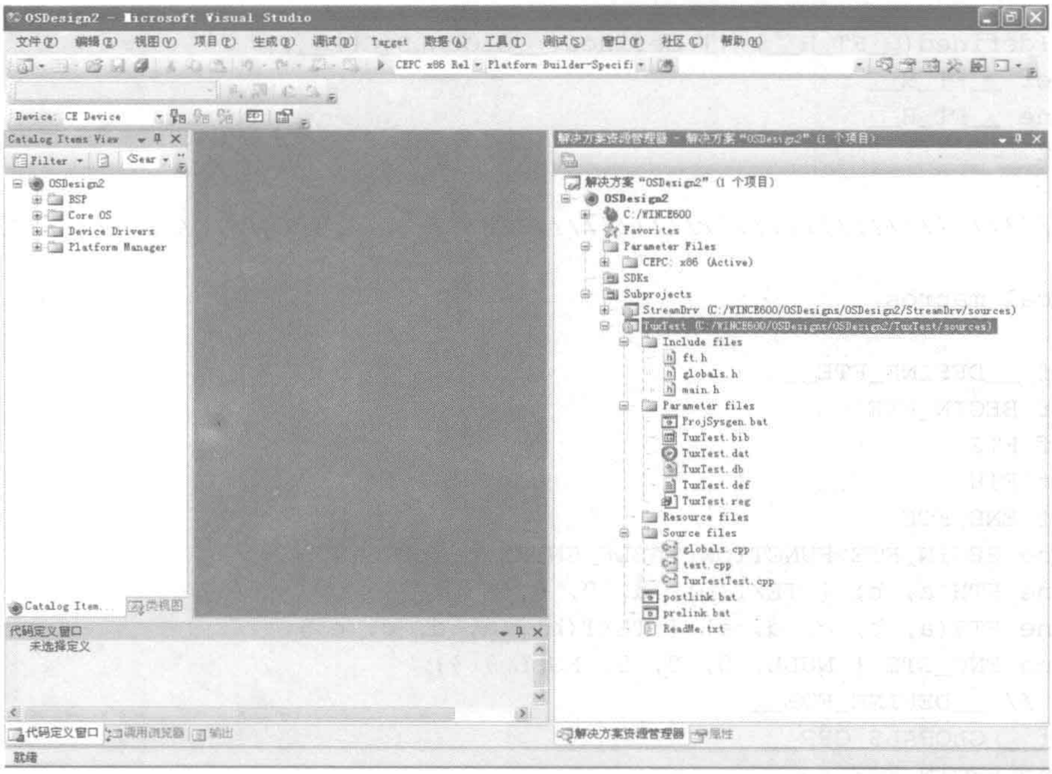


图 9-7

然后手工添加以下文件。

ft.h 文件，其内容为：

```
//
// Copyright (c) Microsoft Corporation.  All rights reserved.
//
//
// Use of this source code is subject to the terms of the Microsoft end-user
// license agreement (EULA) under which you licensed this SOFTWARE PRODUCT.
// If you did not accept the terms of the EULA, you are not authorized to use
// this source code. For a copy of the EULA, please see the LICENSE.RTF on your
// install media.
//
////////////////////////////////////////////////////////////////////////////
//////
//
//  TuxTest TUX DLL
//
//  Module: ft.h
//          Declares the TUX function table and test function prototypes EXCEPT
```

```
    //          when included by globals.cpp, in which case it DEFINES the function
    //          table.
    //
    //  Revision History:
    //
    ////////////////////////////////////////////////////////////////////////////
//////

    #if (!defined(__FT_H__) || defined(__GLOBALS_CPP__))
    #ifndef __FT_H__
    #define __FT_H__
    #endif

    ////////////////////////////////////////////////////////////////////////////
//////
    // Local macros

    #ifdef __DEFINE_FTE__
    #undef BEGIN_FTE
    #undef FTE
    #undef FTH
    #undef END_FTE
    #define BEGIN_FTE FUNCTION_TABLE_ENTRY g_lpFTE[] = {
    #define FTH(a, b) { TEXT(b), a, 0, 0, NULL },
    #define FTE(a, b, c, d, e) { TEXT(b), a, d, c, e },
    #define END_FTE { NULL, 0, 0, 0, NULL } };
    #else // __DEFINE_FTE__
    #ifdef __GLOBALS_CPP__
    #define BEGIN_FTE
    #else // __GLOBALS_CPP__
    #define BEGIN_FTE extern FUNCTION_TABLE_ENTRY g_lpFTE[];
    #endif // __GLOBALS_CPP__
    #define FTH(a, b)
    #define FTE(a, b, c, d, e) TESTPROCAPI e(UINT, TPPARAM,
LPFUNCTION_TABLE_ENTRY);
    #define END_FTE
    #endif // __DEFINE_FTE__

    ////////////////////////////////////////////////////////////////////////////
//////
    // TUX Function table
    //  To create the function table and function prototypes, two macros are
    //  available:
    //
    //      FTH(level, description)
    //          (Function Table Header) Used for entries that don't have functions,
    //          entered only as headers (or comments) into the function table.
    //
    //      FTE(level, description, code, param, function)
    //          (Function Table Entry) Used for all functions. DON'T use this macro
    //          if the "function" field is NULL. In that case, use the FTH macro.
    //
```

```
// You must not use the TEXT or _T macros here. This is done by the FTH and
FTE
// macros.
//
// In addition, the table must be enclosed by the BEGIN_FTE and END_FTE macros.

BEGIN_FTE
    FTH(0, "Sample test cases")
    FTE(1,      "Sample test", 1, 0, TestProc)
END_FTE

/////////////////////////////////////////////////////////////////////////////
//////

#endif // !__FT_H__ || __GLOBALS_CPP__
/////////////////////////////////////////////////////////////////////////////
//////
//
// TuxTest TUX DLL
// Copyright (c) Microsoft Corporation
//
// Module: globals.h
//          Declares all global variables and test function prototypes EXCEPT
//          when included by globals.cpp, in which case it DEFINES global
//          variables, including the function table.
//
// Revision History:
//
/////////////////////////////////////////////////////////////////////////////
//////

#ifndef __GLOBALS_H__
#define __GLOBALS_H__

/////////////////////////////////////////////////////////////////////////////
//////
// Local macros

#ifdef __GLOBALS_CPP__
#define GLOBAL
#define INIT(x) = x
#else // __GLOBALS_CPP__
#define GLOBAL  extern
#define INIT(x)
#endif // __GLOBALS_CPP__

/////////////////////////////////////////////////////////////////////////////
//////
// Global macros

#define countof(x)  (sizeof(x)/sizeof(*(x)))
```

```
//////////////////////////////////////////////////////////////////////////////
//////
  // Global function prototypes

  void          Debug(LPCTSTR, ...);
  SHELLPROCAPI    ShellProc(UINT, SPPARAM);

  //////////////////////////////////////////////////////////////////////////////
//////
  // TUX Function table

  #include "ft.h"

  //////////////////////////////////////////////////////////////////////////////
//////
  // Globals

  // Global CKato logging object. Set while processing SPM_LOAD_DLL message.
  GLOBAL CKato              *g_pKato INIT(NULL);

  // Global shell info structure. Set while processing SPM_SHELL_INFO message.
  GLOBAL SPS_SHELL_INFO    *g_pShellInfo;

  // Add more globals of your own here. There are two macros available for this:
  //  GLOBAL  Precede each declaration/definition with this macro.
  //  INIT    Use this macro to initialize globals, instead of typing "= ..."
  //
  // For example, to declare two DWORDs, one uninitialized and the other
  // initialized to 0x80000000, you could enter the following code:
  //
  //  GLOBAL DWORD          g_dwUninit,
  //                        g_dwInit INIT(0x80000000);
  //////////////////////////////////////////////////////////////////////////////
//////

  #endif // __GLOBALS_H__
  //////////////////////////////////////////////////////////////////////////////
//////
  //
  //  TuxTest TUX DLL
  //  Copyright (c) Microsoft Corporation
  //
  //  Module: main.h
  //          Header for all files in the project.
  //
  //  Revision History:
  //
  //////////////////////////////////////////////////////////////////////////////
//////

  #ifndef __MAIN_H__
  #define __MAIN_H__
```

```
//////////////////////////////////////////////////////////////////////////////
//////
// Included files

#include <windows.h>
#include <stdlib.h>
#include <tchar.h>
#include <tux.h>
#include <kato.h>

//////////////////////////////////////////////////////////////////////////////
//////
// Suggested log verbosities

#define LOG_EXCEPTION          0
#define LOG_FAIL               2
#define LOG_ABORT              4
#define LOG_SKIP               6
#define LOG_NOT_IMPLEMENTED    8
#define LOG_PASS              10
#define LOG_DETAIL            12
#define LOG_COMMENT           14

//////////////////////////////////////////////////////////////////////////////
//////

#endif // __MAIN_H__
```

globals.h 文件，其内容为：

```
//
// Copyright (c) Microsoft Corporation.  All rights reserved.
//
//
// Use of this source code is subject to the terms of the Microsoft end-user
// license agreement (EULA) under which you licensed this SOFTWARE PRODUCT.
// If you did not accept the terms of the EULA, you are not authorized to use
// this source code. For a copy of the EULA, please see the LICENSE.RTF on your
// install media.
//
//////////////////////////////////////////////////////////////////////////////
//////
//
//  TuxTest TUX DLL
//
//  Module: globals.h
//          Declares all global variables and test function prototypes EXCEPT
//          when included by globals.cpp, in which case it DEFINES global
//          variables, including the function table.
//
//  Revision History:
```

```
//
////////////////////////////////////////////////////////////////////////////
//////

#ifndef __GLOBALS_H__
#define __GLOBALS_H__

////////////////////////////////////////////////////////////////////////////
//////
// Local macros

#ifdef __GLOBALS_CPP__
#define GLOBAL
#define INIT(x) = x
#else // __GLOBALS_CPP__
#define GLOBAL  extern
#define INIT(x)
#endif // __GLOBALS_CPP__

////////////////////////////////////////////////////////////////////////////
//////
// Global macros

#define countof(x)  (sizeof(x)/sizeof(*(x)))

////////////////////////////////////////////////////////////////////////////
//////
// Global function prototypes

void            Debug(LPCTSTR, ...);
SHELLPROCAPI    ShellProc(UINT, SPPARAM);

////////////////////////////////////////////////////////////////////////////
//////
// TUX Function table

#include "ft.h"

////////////////////////////////////////////////////////////////////////////
//////
// Globals

// Global CKato logging object. Set while processing SPM_LOAD_DLL message.
GLOBAL CKato            *g_pKato INIT(NULL);

// Global shell info structure. Set while processing SPM_SHELL_INFO message.
GLOBAL SPS_SHELL_INFO   *g_pShellInfo;

// Add more globals of your own here. There are two macros available for this:
//  GLOBAL  Precede each declaration/definition with this macro.
//  INIT    Use this macro to initialize globals, instead of typing "= ..."
//
```

```
// For example, to declare two DWORDs, one uninitialized and the other
// initialized to 0x80000000, you could enter the following code:
//
//  GLOBAL DWORD          g_dwUninit,
//                        g_dwInit INIT(0x80000000);
//////////////////////////////////////////////////////////////////////////
//////

#endif // __GLOBALS_H__
```

main.h 文件，其内容为：

```
//
// Copyright (c) Microsoft Corporation.  All rights reserved.
//
//
// Use of this source code is subject to the terms of the Microsoft end-user
// license agreement (EULA) under which you licensed this SOFTWARE PRODUCT.
// If you did not accept the terms of the EULA, you are not authorized to use
// this source code. For a copy of the EULA, please see the LICENSE.RTF on your
// install media.
//
//////////////////////////////////////////////////////////////////////////
//////
//
//  TuxTest TUX DLL
//
//  Module: main.h
//          Header for all files in the project.
//
//  Revision History:
//
//////////////////////////////////////////////////////////////////////////
//////

#ifndef __MAIN_H__
#define __MAIN_H__

//////////////////////////////////////////////////////////////////////////
//////
// Included files

#include <windows.h>
#include <stdlib.h>
#include <tchar.h>
#include <tux.h>
#include <kato.h>

//////////////////////////////////////////////////////////////////////////
```

```
//////
   // Suggested log verbosities

   #define LOG_EXCEPTION          0
   #define LOG_FAIL               2
   #define LOG_ABORT              4
   #define LOG_SKIP               6
   #define LOG_NOT_IMPLEMENTED    8
   #define LOG_PASS              10
   #define LOG_DETAIL            12
   #define LOG_COMMENT           14

   //////////////////////////////////////////////////////////////////////////
//////

   #endif // __MAIN_H__
```

globals.cpp 文件，其内容为：

```
   //
   // Copyright (c) Microsoft Corporation.  All rights reserved.
   //
   //
   // Use of this source code is subject to the terms of the Microsoft end-user
   // license agreement (EULA) under which you licensed this SOFTWARE PRODUCT.
   // If you did not accept the terms of the EULA, you are not authorized to use
   // this source code. For a copy of the EULA, please see the LICENSE.RTF on your
   // install media.
   //
   //////////////////////////////////////////////////////////////////////////
//////
   //
   //  TuxTest TUX DLL
   //
   //  Module: globals.cpp
   //          Causes the header globals.h to actually define the global
variables.
   //
   //  Revision History:
   //
   //////////////////////////////////////////////////////////////////////////
//////

   #include "main.h"

   // By defining the symbol __GLOBALS_CPP__, we force the file globals.h to
   // define, instead of declare, all global variables.
   #define __GLOBALS_CPP__
   #include "globals.h"
```

```
// By defining the symbol __DEFINE_FTE__, we force the file ft.h to define
#define __DEFINE_FTE__
#include "ft.h"
```

Test.cpp 文件，其内容为：

```
//
// Copyright (c) Microsoft Corporation.  All rights reserved.
//
//
// Use of this source code is subject to the terms of the Microsoft end-user
// license agreement (EULA) under which you licensed this SOFTWARE PRODUCT.
// If you did not accept the terms of the EULA, you are not authorized to use
// this source code. For a copy of the EULA, please see the LICENSE.RTF on your
// install media.
//
////////////////////////////////////////////////////////////////////////////////
//////
//
//  TuxTest TUX DLL
//
//  Module: test.cpp
//          Contains the test functions.
//
//  Revision History:
//
////////////////////////////////////////////////////////////////////////////////
//////

#include "main.h"
#include "globals.h"

////////////////////////////////////////////////////////////////////////////////
//////
// TestProc
//  Executes one test.
//
// Parameters:
//  uMsg            Message code.
//  tpParam         Additional message-dependent data.
//  lpFTE           Function table entry that generated this call.
//
// Return value:
//  TPR_PASS if the test passed, TPR_FAIL if the test fails, or possibly other
//  special conditions.

TESTPROCAPI TestProc(UINT uMsg, TPPARAM tpParam, LPFUNCTION_TABLE_ENTRY
```

```
lpFTE)
   {
   TESTPROCAPI tRet=TPR_PASS; // Assume the Test has Passed OK - Fail if we find
any problems.
   HINSTANCE hInstDLL;
   FARPROC fnPointer;

      // The shell doesn't necessarily want us to execute the test. Make sure
      // first.
      if(uMsg != TPM_EXECUTE)
      {
         return TPR_NOT_HANDLED;
      }
      g_pKato->Log(LOG_COMMENT, TEXT("StreamDrv Test Starting"));

      g_pKato->Log(LOG_COMMENT, TEXT("SteamDrv Test - Loading StreamDrv.DLL"));
   hInstDLL=LoadLibrary(TEXT("StreamDrv.dll"));
   if (NULL == hInstDLL) {
      g_pKato->Log(LOG_COMMENT, TEXT("SteamDrv Test - Failed to load
StreamDrv.DLL"));
      tRet=TPR_FAIL;
   } else {
      g_pKato->Log(LOG_COMMENT, TEXT("SteamDrv Test - Loaded StreamDrv.DLL
OK"));
   // Now check the Driver interfaces...
      g_pKato->Log(LOG_COMMENT, TEXT("SteamDrv Test - Checking
Interfaces..."));

   // Open
      g_pKato->Log(LOG_COMMENT, TEXT("SteamDrv Test - Checking DEM_Open"));
      fnPointer=GetProcAddress(hInstDLL,TEXT("DEM_Open"));
      if (NULL == fnPointer) {
         g_pKato->Log(LOG_COMMENT, TEXT("SteamDrv Test - DEM_Open not
found"));
         tRet=TPR_FAIL;
      } else {
         g_pKato->Log(LOG_COMMENT, TEXT("SteamDrv Test - DEM_Open OK"));
      }

   // Close
      g_pKato->Log(LOG_COMMENT, TEXT("SteamDrv Test - Checking DEM_Close"));
      fnPointer=GetProcAddress(hInstDLL,TEXT("DEM_Close"));
      if (NULL == fnPointer) {
         g_pKato->Log(LOG_COMMENT, TEXT("SteamDrv Test - DEM_Close not
found"));
         tRet=TPR_FAIL;
      } else {
```

```
        g_pKato->Log(LOG_COMMENT, TEXT("SteamDrv Test - DEM_Close OK"));
    }

  // Read
    g_pKato->Log(LOG_COMMENT, TEXT("SteamDrv Test - Checking DEM_Read"));
    fnPointer=GetProcAddress(hInstDLL,TEXT("DEM_Read"));
    if (NULL == fnPointer) {
        g_pKato->Log(LOG_COMMENT, TEXT("SteamDrv Test - DEM_Read not
found"));
        tRet=TPR_FAIL;
    } else {
        g_pKato->Log(LOG_COMMENT, TEXT("SteamDrv Test - DEM_Read OK"));
    }

  // Write
    g_pKato->Log(LOG_COMMENT, TEXT("SteamDrv Test - Checking DEM_Write"));
    fnPointer=GetProcAddress(hInstDLL,TEXT("DEM_Write"));
    if (NULL == fnPointer) {
        g_pKato->Log(LOG_COMMENT, TEXT("SteamDrv Test - DEM_Write not
found"));
        tRet=TPR_FAIL;
    } else {
        g_pKato->Log(LOG_COMMENT, TEXT("SteamDrv Test - DEM_Write OK"));
    }

  // Init
    g_pKato->Log(LOG_COMMENT, TEXT("SteamDrv Test - Checking DEM_Init"));
    fnPointer=GetProcAddress(hInstDLL,TEXT("DEM_Init"));
    if (NULL == fnPointer) {
        g_pKato->Log(LOG_COMMENT, TEXT("SteamDrv Test - DEM_Init not
found"));
        tRet=TPR_FAIL;
    } else {
        g_pKato->Log(LOG_COMMENT, TEXT("SteamDrv Test - DEM_Init OK"));
    }

  // Deinit
    g_pKato->Log(LOG_COMMENT, TEXT("SteamDrv Test - Checking DEM_Deinit"));
    fnPointer=GetProcAddress(hInstDLL,TEXT("DEM_Deinit"));
    if (NULL == fnPointer) {
        g_pKato->Log(LOG_COMMENT, TEXT("SteamDrv Test - DEM_Deinit not
found"));
        tRet=TPR_FAIL;
    } else {
        g_pKato->Log(LOG_COMMENT, TEXT("SteamDrv Test - DEM_Deinit OK"));
    }

  // PowerUp
```

```
        g_pKato->Log(LOG_COMMENT, TEXT("SteamDrv Test - Checking DEM_PowerUp"));
        fnPointer=GetProcAddress(hInstDLL,TEXT("DEM_PowerUp"));
        if (NULL == fnPointer) {
            g_pKato->Log(LOG_COMMENT, TEXT("SteamDrv Test - DEM_PowerUp not
found"));
            tRet=TPR_FAIL;
        } else {
            g_pKato->Log(LOG_COMMENT, TEXT("SteamDrv Test - DEM_PowerUp OK"));
        }

    // PowerDown
        g_pKato->Log(LOG_COMMENT, TEXT("SteamDrv Test - Checking
DEM_PowerDown"));
        fnPointer=GetProcAddress(hInstDLL,TEXT("DEM_PowerDown"));
        if (NULL == fnPointer) {
            g_pKato->Log(LOG_COMMENT, TEXT("SteamDrv Test - DEM_PowerDown not
found"));
            tRet=TPR_FAIL;
        } else {
            g_pKato->Log(LOG_COMMENT, TEXT("SteamDrv Test - DEM_PowerDown OK"));
        }

    // IOControl
        g_pKato->Log(LOG_COMMENT, TEXT("SteamDrv Test - Checking
DEM_IOControl"));
        fnPointer=GetProcAddress(hInstDLL,TEXT("DEM_IOControl"));
        if (NULL == fnPointer) {
            g_pKato->Log(LOG_COMMENT, TEXT("SteamDrv Test - DEM_IOControl not
found"));
            tRet=TPR_FAIL;
        } else {
            g_pKato->Log(LOG_COMMENT, TEXT("SteamDrv Test - DEM_IOControl OK"));
        }

    // Seek
        g_pKato->Log(LOG_COMMENT, TEXT("SteamDrv Test - Checking DEM_Seek"));
        fnPointer=GetProcAddress(hInstDLL,TEXT("DEM_Seek"));
        if (NULL == fnPointer) {
            g_pKato->Log(LOG_COMMENT, TEXT("SteamDrv Test - DEM_Seek not
found"));
            tRet=TPR_FAIL;
        } else {
            g_pKato->Log(LOG_COMMENT, TEXT("SteamDrv Test - DEM_Seek OK"));
        }
        FreeLibrary(hInstDLL);
    }
```

```
    return tRet;
}

//////////////////////////////////////////////////////////////////////////
//////
```

TuxTestTest.cpp 文件，其内容为：

```
//
// Copyright (c) Microsoft Corporation.  All rights reserved.
//
//
// Use of this source code is subject to the terms of the Microsoft end-user
// license agreement (EULA) under which you licensed this SOFTWARE PRODUCT.
// If you did not accept the terms of the EULA, you are not authorized to use
// this source code. For a copy of the EULA, please see the LICENSE.RTF on your
// install media.
//
//////////////////////////////////////////////////////////////////////////
//////
//
//  TuxTest TUX DLL
//
//  Module: TuxTestTest.cpp
//          Contains the shell processing function.
//
//  Revision History:
//
//////////////////////////////////////////////////////////////////////////
//////

#include "main.h"
#include "globals.h"

//////////////////////////////////////////////////////////////////////////
//////
// DllMain
//  Main entry point of the DLL. Called when the DLL is loaded or unloaded.
//
// Parameters:
//  hInstance       Module instance of the DLL.
//  dwReason        Reason for the function call.
//  lpReserved      Reserved for future use.
//
// Return value:
//  TRUE if successful, FALSE to indicate an error condition.

BOOL WINAPI DllMain(HANDLE hInstance, ULONG dwReason, LPVOID lpReserved)
```

```
{
    // Any initialization code goes here.

    return TRUE;
}

////////////////////////////////////////////////////////////////////////////
//////
// Debug
//  Printf-like wrapping around OutputDebugString.
//
// Parameters:
//  szFormat        Formatting string (as in printf).
//  ...             Additional arguments.
//
// Return value:
//  None.

void Debug(LPCTSTR szFormat, ...)
{
    static  TCHAR   szHeader[] = TEXT("TuxTest: ");
    TCHAR   szBuffer[1024];

    va_list pArgs;
    va_start(pArgs, szFormat);
    lstrcpy(szBuffer, szHeader);
    wvsprintf(
        szBuffer + countof(szHeader) - 1,
        szFormat,
        pArgs);
    va_end(pArgs);

    _tcscat(szBuffer, TEXT("\r\n"));

    OutputDebugString(szBuffer);
}

////////////////////////////////////////////////////////////////////////////
//////
// ShellProc
//  Processes messages from the TUX shell.
//
// Parameters:
//  uMsg            Message code.
//  spParam         Additional message-dependent data.
//
// Return value:
//  Depends on the message.
```

```
SHELLPROCAPI ShellProc(UINT uMsg, SPPARAM spParam)
{
    LPSPS_BEGIN_TEST    pBT;
    LPSPS_END_TEST      pET;

    switch (uMsg)
    {
    case SPM_LOAD_DLL:
        // Sent once to the DLL immediately after it is loaded. The spParam
        // parameter will contain a pointer to a SPS_LOAD_DLL structure. The
        // DLL should set the fUnicode member of this structure to TRUE if the
        // DLL is built with the UNICODE flag set. If you set this flag, Tux
        // will ensure that all strings passed to your DLL will be in UNICODE
        // format, and all strings within your function table will be processed
        // by Tux as UNICODE. The DLL may return SPR_FAIL to prevent the DLL
        // from continuing to load.
        Debug(TEXT("ShellProc(SPM_LOAD_DLL, ...) called"));

        // If we are UNICODE, then tell Tux this by setting the following flag.
#ifdef UNICODE
        ((LPSPS_LOAD_DLL)spParam)->fUnicode = TRUE;
#endif // UNICODE
        g_pKato = (CKato*)KatoGetDefaultObject();
        break;

    case SPM_UNLOAD_DLL:
        // Sent once to the DLL immediately before it is unloaded.
        Debug(TEXT("ShellProc(SPM_UNLOAD_DLL, ...) called"));
        break;

    case SPM_SHELL_INFO:
        // Sent once to the DLL immediately after SPM_LOAD_DLL to give the DLL
        // some useful information about its parent shell and environment. The
        // spParam parameter will contain a pointer to a SPS_SHELL_INFO
        // structure. The pointer to the structure may be stored for later use
        // as it will remain valid for the life of this Tux Dll. The DLL may
        // return SPR_FAIL to prevent the DLL from continuing to load.
        Debug(TEXT("ShellProc(SPM_SHELL_INFO, ...) called"));

        // Store a pointer to our shell info for later use.
        g_pShellInfo = (LPSPS_SHELL_INFO)spParam;
        break;

    case SPM_REGISTER:
        // This is the only ShellProc() message that a DLL is required to
        // handle (except for SPM_LOAD_DLL if you are UNICODE). This message
is
```

```
            // sent once to the DLL immediately after the SPM_SHELL_INFO message
to
            // query the DLL for its function table. The spParam will contain a
            // pointer to a SPS_REGISTER structure. The DLL should store its
            // function table in the lpFunctionTable member of the SPS_REGISTER
            // structure. The DLL may return SPR_FAIL to prevent the DLL from
            // continuing to load.
            Debug(TEXT("ShellProc(SPM_REGISTER, ...) called"));
            ((LPSPS_REGISTER)spParam)->lpFunctionTable = g_lpFTE;
    #ifdef UNICODE
            return SPR_HANDLED | SPF_UNICODE;
    #else // UNICODE
            return SPR_HANDLED;
    #endif // UNICODE

        case SPM_START_SCRIPT:
            // Sent to the DLL immediately before a script is started. It is sent
            // to all Tux DLLs, including loaded Tux DLLs that are not in the
            // script. All DLLs will receive this message before the first
            // TestProc() in the script is called.
            Debug(TEXT("ShellProc(SPM_START_SCRIPT, ...) called"));
            break;

        case SPM_STOP_SCRIPT:
            // Sent to the DLL when the script has stopped. This message is sent
            // when the script reaches its end, or because the user pressed
            // stopped prior to the end of the script. This message is sent to
            // all Tux DLLs, including loaded Tux DLLs that are not in the script.
            Debug(TEXT("ShellProc(SPM_STOP_SCRIPT, ...) called"));
            break;

        case SPM_BEGIN_GROUP:
            // Sent to the DLL before a group of tests from that DLL is about to
            // be executed. This gives the DLL a time to initialize or allocate
            // data for the tests to follow. Only the DLL that is next to run
            // receives this message. The prior DLL, if any, will first receive
            // a SPM_END_GROUP message. For global initialization and
            // de-initialization, the DLL should probably use SPM_START_SCRIPT
            // and SPM_STOP_SCRIPT, or even SPM_LOAD_DLL and SPM_UNLOAD_DLL.
            Debug(TEXT("ShellProc(SPM_BEGIN_GROUP, ...) called"));
            g_pKato->BeginLevel(0, TEXT("BEGIN GROUP: TuxTest.DLL"));
            break;

        case SPM_END_GROUP:
            // Sent to the DLL after a group of tests from that DLL has completed
            // running. This gives the DLL a time to cleanup after it has been
            // run. This message does not mean that the DLL will not be called
            // again; it just means that the next test to run belongs to a
```

```
        // different DLL. SPM_BEGIN_GROUP and SPM_END_GROUP allow the DLL
        // to track when it is active and when it is not active.
        Debug(TEXT("ShellProc(SPM_END_GROUP, ...) called"));
        g_pKato->EndLevel(TEXT("END GROUP: TuxTest.DLL"));
        break;

    case SPM_BEGIN_TEST:
        // Sent to the DLL immediately before a test executes. This gives
        // the DLL a chance to perform any common action that occurs at the
        // beginning of each test, such as entering a new logging level.
        // The spParam parameter will contain a pointer to a SPS_BEGIN_TEST
        // structure, which contains the function table entry and some other
        // useful information for the next test to execute. If the ShellProc
        // function returns SPR_SKIP, then the test case will not execute.
        Debug(TEXT("ShellProc(SPM_BEGIN_TEST, ...) called"));
        // Start our logging level.
        pBT = (LPSPS_BEGIN_TEST)spParam;
        g_pKato->BeginLevel(
            pBT->lpFTE->dwUniqueID,
            TEXT("BEGIN TEST: \"%s\", Threads=%u, Seed=%u"),
            pBT->lpFTE->lpDescription,
            pBT->dwThreadCount,
            pBT->dwRandomSeed);
        break;

    case SPM_END_TEST:
        // Sent to the DLL after a single test executes from the DLL.
        // This gives the DLL a time perform any common action that occurs at
        // the completion of each test, such as exiting the current logging
        // level. The spParam parameter will contain a pointer to a
        // SPS_END_TEST structure, which contains the function table entry and
        // some other useful information for the test that just completed. If
        // the ShellProc returned SPR_SKIP for a given test case, then the
        // ShellProc() will not receive a SPM_END_TEST for that given test case.
        Debug(TEXT("ShellProc(SPM_END_TEST, ...) called"));
        // End our logging level.
        pET = (LPSPS_END_TEST)spParam;
        g_pKato->EndLevel(
            TEXT("END TEST: \"%s\", %s, Time=%u.%03u"),
            pET->lpFTE->lpDescription,
            pET->dwResult == TPR_SKIP ? TEXT("SKIPPED") :
            pET->dwResult == TPR_PASS ? TEXT("PASSED") :
            pET->dwResult == TPR_FAIL ? TEXT("FAILED") : TEXT("ABORTED"),
            pET->dwExecutionTime / 1000, pET->dwExecutionTime % 1000);
        break;

    case SPM_EXCEPTION:
        // Sent to the DLL whenever code execution in the DLL causes and
```

```
            // exception fault. TUX traps all exceptions that occur while
            // executing code inside a test DLL.
            Debug(TEXT("ShellProc(SPM_EXCEPTION, ...) called"));
            g_pKato->Log(LOG_EXCEPTION, TEXT("Exception occurred!"));
            break;

        default:
            // Any messages that we haven't processed must, by default, cause us
            // to return SPR_NOT_HANDLED. This preserves compatibility with future
            // versions of the TUX shell protocol, even if new messages are added.
            return SPR_NOT_HANDLED;
        }
        return SPR_HANDLED;
    }
```

//

第 10 章　EVC 程序开发

在 Windows CE 平台上，EVC 曾经是最好的应用程序开发工具。如今，尽管微软不再提供 EVC 的独立发行版本，而是将它并入了 VS2005。但丝毫不妨碍广大程序开发人员继续使用 EVC4.0 版本。本章将介绍 EVC 的基本使用方法以及用 EVC 开发一个串口通信的 DLL。

10.1　EVC 应用程序开发入门

步骤 1　如图 10-1 所示，打开 eMbedded VC++，选择新建应用程序，选择 WCE Application，在项目名称处输入 HELLOCE，检查 CPUS 是否选中了 ARM 和 X86。

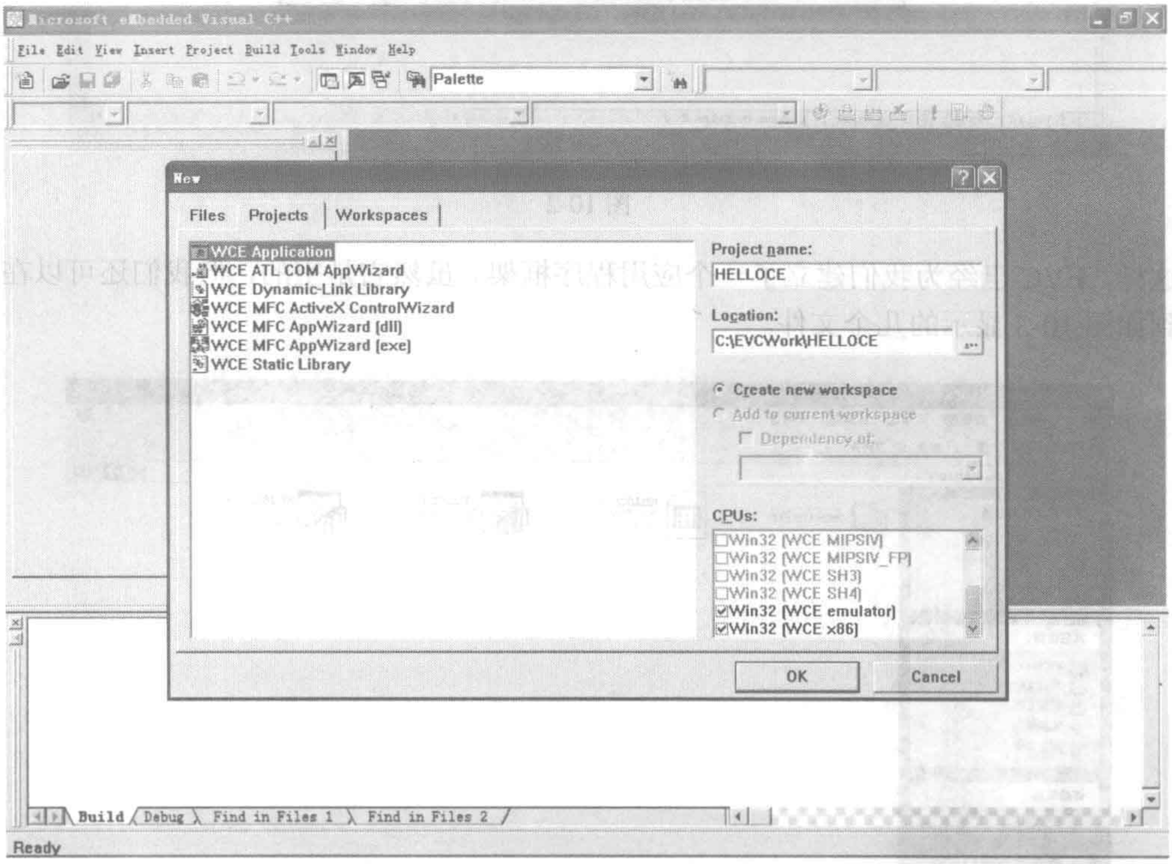


图 10-1

步骤 2　如图 10-2 所示，选择“An empty Project”，建立一个空项目，然后单击“Finish”按钮。

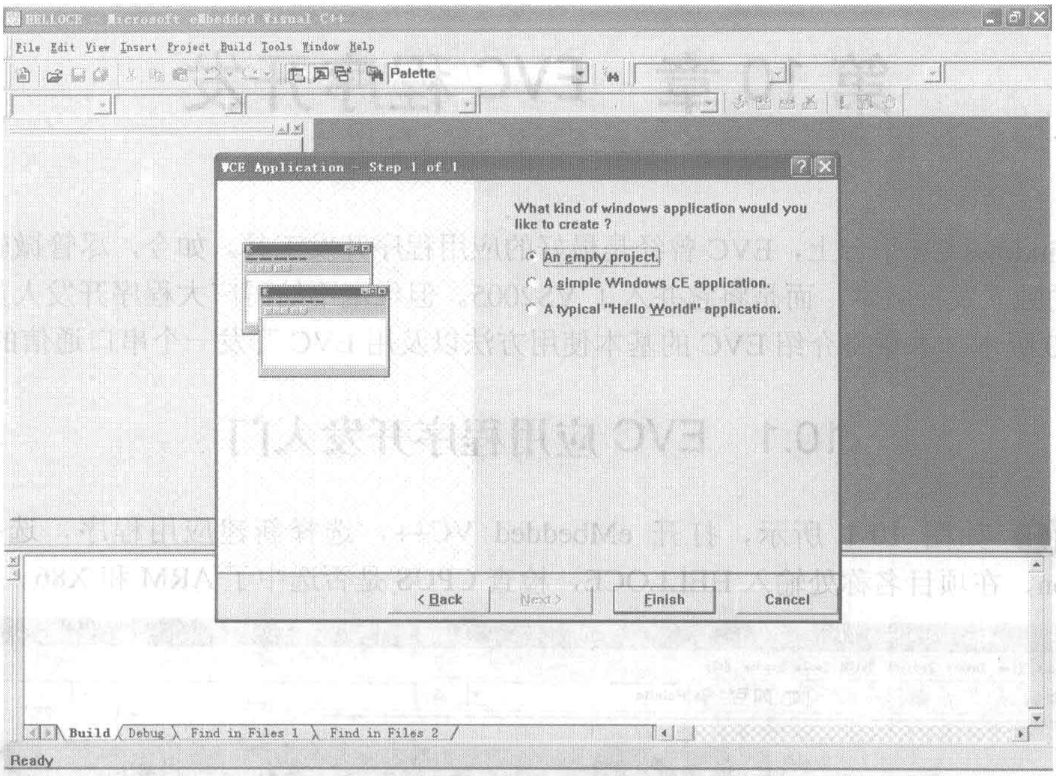


图 10-2

这样，EVC 已经为我们建立了一个应用程序框架。虽然它是空的，但我们还可以在目录下找到如图 10-3 显示的几个文件。

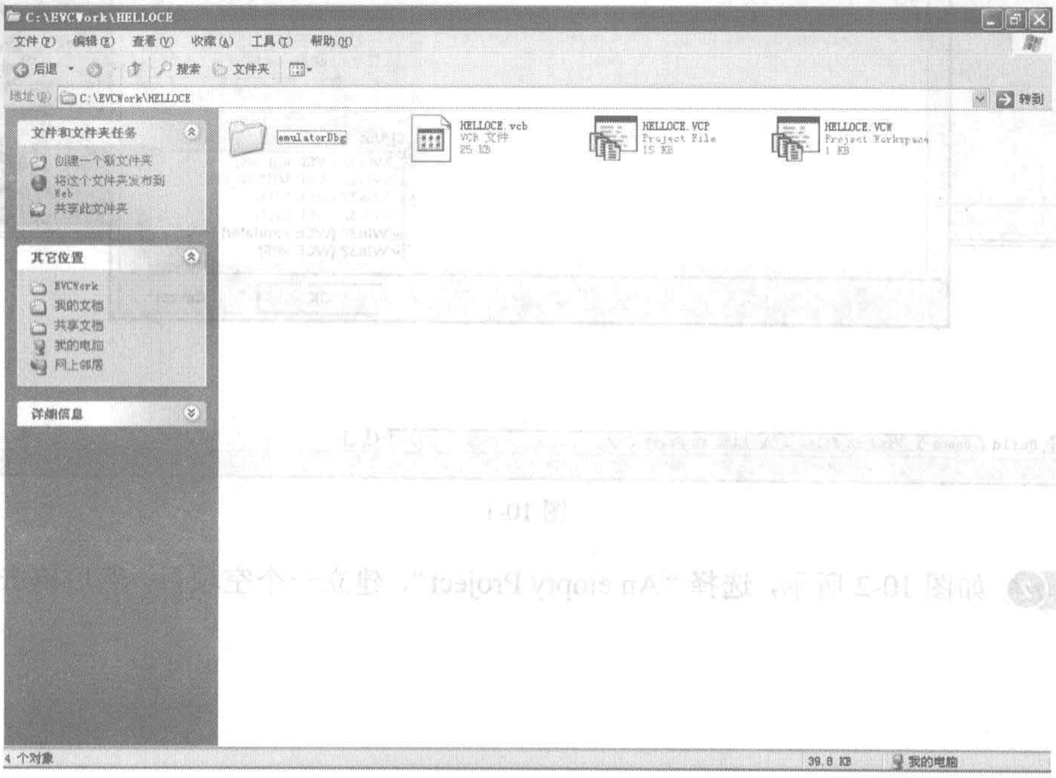


图 10-3

接下来向项目中添加一个头文件 HelloCE.h 和一个 HelloCE.c 文件。

文件内容如下。

```
//======================================================================
// Header file: helloce.h
//======================================================================
// 返回元素的数量，主要用于搜索消息列表
#define dim(x) (sizeof(x) / sizeof(x[0]))
//----------------------------------------------------------------------
//数据类型定义
//
struct decodeUINT { //消息和消息函数的关联结构
 UINT Code;
 LRESULT (*Fxn)(HWND, UINT, WPARAM, LPARAM); //这里用到了函数指针
};
struct decodeCMD { //菜单和处理函数的关联结构
 UINT Code;
 LRESULT (*Fxn)(HWND, WORD, HWND, WORD); //这里用到了函数指针
};

//----------------------------------------------------------------------
#define IDC_CMDBAR 1 // 命令条ID

//----------------------------------------------------------------------
// 函数原型
//
int InitApp (HINSTANCE); //初始化应用函数原型
HWND InitInstance (HINSTANCE, LPWSTR, int); //初始化实例函数原型
int TermInstance (HINSTANCE, int); //实例终止函数原型
// 窗口处理函数原型
LRESULT CALLBACK MainWndProc (HWND, UINT, WPARAM, LPARAM);
// 消息句柄
LRESULT DoCreateMain (HWND, UINT, WPARAM, LPARAM);
LRESULT DoPaintMain (HWND, UINT, WPARAM, LPARAM);
LRESULT DoHibernateMain (HWND, UINT, WPARAM, LPARAM);
LRESULT DoActivateMain (HWND, UINT, WPARAM, LPARAM);
LRESULT DoDestroyMain (HWND, UINT, WPARAM, LPARAM);

//======================================================================
// HelloCE - helloce.c
//======================================================================
#include <windows.h>
#include <commctrl.h>
#include "helloce.h"
//----------------------------------------------------------------------
// 全局数据
//
const TCHAR szAppName[] = TEXT ("HelloCE");
HINSTANCE hInst; // 程序的实例句柄
//主窗口过程函数的消息映射表用到decodeUINT结构
```

```
const struct decodeUINT MainMessages[] = {
  WM_CREATE, DoCreateMain,
  WM_PAINT, DoPaintMain,
  WM_HIBERNATE, DoHibernateMain,
  WM_ACTIVATE, DoActivateMain,
  WM_DESTROY, DoDestroyMain,
};

//======================================================================
// 程序的入口
int WINAPI WinMain (HINSTANCE hInstance, HINSTANCE hPrevInstance,
LPWSTR lpCmdLine, int nCmdShow) {
  MSG msg;
  int rc = 0;
  HWND hwndMain;
  // 初始应用
  rc = InitApp (hInstance);
  if (rc) return rc;
    // 初始化实例
    hwndMain = InitInstance (hInstance, lpCmdLine, nCmdShow);
    if (hwndMain == 0)
      return 0x10;
    // 应用程序消息循环
    while (GetMessage (&msg, NULL, 0, 0)) {
      TranslateMessage (&msg);
      DispatchMessage (&msg);
    }
    //实例清除
    return TermInstance (hInstance, msg.wParam);
  }
  //----------------------------------------------------------------------
  // 应用程序初始化函数
  //
  int InitApp (HINSTANCE hInstance) {
    WNDCLASS wc;
    //注册应用程序的主窗口类
    wc.style = 0; // 窗口样式
    wc.lpfnWndProc = MainWndProc; // 回调函数
    wc.cbClsExtra = 0; // 扩展的类数据
    wc.cbWndExtra = 0; // 扩展的窗口数据
    wc.hInstance = hInstance; //实例句柄
    wc.hIcon = NULL, // 图标
    wc.hCursor = NULL; // 鼠标
    wc.hbrBackground = (HBRUSH) GetStockObject (WHITE_BRUSH);
    wc.lpszMenuName = NULL; //菜单
    wc.lpszClassName = szAppName; //窗口类的名字

    if (RegisterClass (&wc) == 0) return 1;
      return 0;
    }
    //----------------------------------------------------------------------
    //初始化实例
```

```
//
HWND InitInstance (HINSTANCE hInstance, LPWSTR lpCmdLine,
int nCmdShow) {
  HWND hWnd;
  // 存储程序实例句柄到全局变量
  hInst = hInstance;
  // 建立主窗口
  hWnd = CreateWindow (szAppName, // 窗口类
    TEXT("Hello CE!"), //窗口标题
    WS_VISIBLE, //样式
    CW_USEDEFAULT, // x坐标
    CW_USEDEFAULT, // y 坐标
    CW_USEDEFAULT, // 初始宽度
    CW_USEDEFAULT, // 初始高度
    NULL, // 父窗口
    NULL, //菜单，必须为NULL，WINCE窗口不支持菜单。
    hInstance, // 实例
    NULL); //建立参数的指针，用于WM_CRATE消息期间。
 // 如果不能建立主窗口返回失败
 if (!IsWindow (hWnd)) return 0;
  // 显示和更新窗口函数
  ShowWindow (hWnd, nCmdShow);
  UpdateWindow (hWnd);
  return hWnd;
 }
//----------------------------------------------------------------------
// TermInstance -程序清除
//
int TermInstance (HINSTANCE hInstance, int nDefRC) {
 return nDefRC;
}
//======================================================================
// 下面是主窗口的消息处理函数
//
//----------------------------------------------------------------------
// MainWndProc - 主过程函数，这是一个回调函数
//
LRESULT CALLBACK MainWndProc (HWND hWnd, UINT wMsg, WPARAM wParam,
LPARAM lParam) {
 INT i;
 //搜索消息列表，如果编写了对应的函数来处理这个消息则调用这个函数
 for (i = 0; i < dim(MainMessages); i++) {
  if (wMsg == MainMessages[i].Code)
   return (*MainMessages[i].Fxn)(hWnd, wMsg, wParam, lParam);
 }
 return DefWindowProc (hWnd, wMsg, wParam, lParam); //没有编写对应的函数则调用默
认的
}
//----------------------------------------------------------------------
// DoCreateMain - 处理窗口建立(WM_CREATE)消息的函数.
//
LRESULT DoCreateMain (HWND hWnd, UINT wMsg, WPARAM wParam,
```

```
LPARAM lParam) {
  HWND hwndCB;
  // 建立命令条.
  hwndCB = CommandBar_Create (hInst, hWnd, IDC_CMDBAR);
  // 添加退出按钮到命令条上
  CommandBar_AddAdornments (hwndCB, 0, 0);
  return 0;
}
//----------------------------------------------------------------------
// DoPaintMain - 处理窗口重画(WM_PAINT)消息的函数
//
LRESULT DoPaintMain (HWND hWnd, UINT wMsg, WPARAM wParam,
LPARAM lParam) {
  PAINTSTRUCT ps;
  RECT rect;
  HDC hdc;
  // 调整客户区域的大小并考虑命令条的高度
  GetClientRect (hWnd, &rect);
  rect.top += CommandBar_Height (GetDlgItem (hWnd, IDC_CMDBAR));
  hdc = BeginPaint (hWnd, &ps);
  DrawText (hdc, TEXT ("你好晕倒的蜥蜴! "), -1, &rect, //被改成了中文
  DT_CENTER | DT_VCENTER | DT_SINGLELINE);
  EndPaint (hWnd, &ps);
  return 0;
}
//----------------------------------------------------------------------
// DoHibernateMain - 处理窗口挂起消息(WM_HIBERNATE)的函数，这是WINCE独有的消息，目的
//是将内存的使用量将到最小.
//
LRESULT DoHibernateMain (HWND hWnd, UINT wMsg, WPARAM wParam, LPARAM lParam) {
  // 如果窗口不是活动的，则取消命令条，释放内存
  if (GetActiveWindow () != hWnd)
    CommandBar_Destroy (GetDlgItem (hWnd, IDC_CMDBAR));
    return 0;
}
//----------------------------------------------------------------------
// DoActivateMain - 处理窗口激活(WM_ACTIVATE)消息的函数
//
LRESULT DoActivateMain (HWND hWnd, UINT wMsg, WPARAM wParam, LPARAM lParam) {
  HWND hwndCB;
  // 如果窗口正处在活动状态而没有命令条则建立它
  if ((LOWORD (wParam) != WA_INACTIVE) && (GetDlgItem (hWnd, IDC_CMDBAR) == 0))
{
    // 建立命令条
    hwndCB = CommandBar_Create (hInst, hWnd, IDC_CMDBAR);
    //添加退出按钮到命令条
    CommandBar_AddAdornments (hwndCB, 0, 0);
  }
  return 0;
}
//----------------------------------------------------------------------
// DoDestroyMain - 处理窗口销毁(WM_DESTROY)消息函数.
```

```
//
LRESULT DoDestroyMain (HWND hWnd, UINT wMsg, WPARAM wParam, LPARAM lParam) {
  PostQuitMessage (0);
  return 0;
}
```

在编译程序之前还需要做一些工作，当然这不是完全必需的，它是需要根据用户的开发环境来进行的一系列设置工作，如图 10-4 所示。

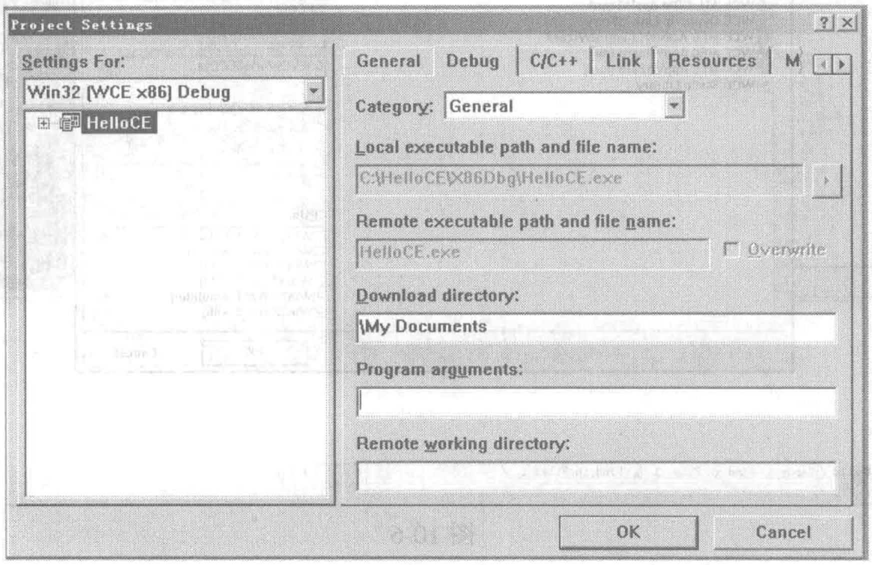


图 10-4

另外一个需要修改的地方是修改 Project Settings 对话框的资源选项卡，将语言选项设为中文，如图 10-5 所示。

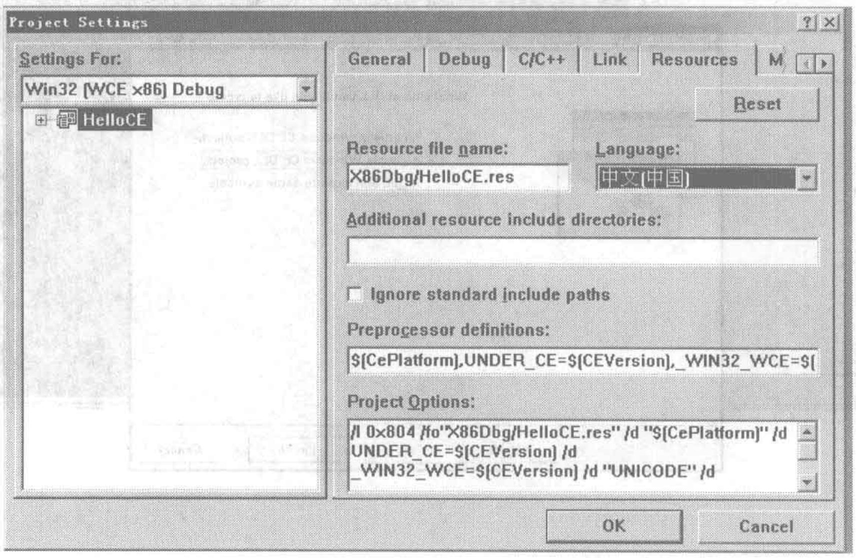


图 10-5

10.2 EVC 开发串口通信 DLL

步骤 1 如图 10-6 所示，打开 EVC，新建工程，选择动态链接库工程。

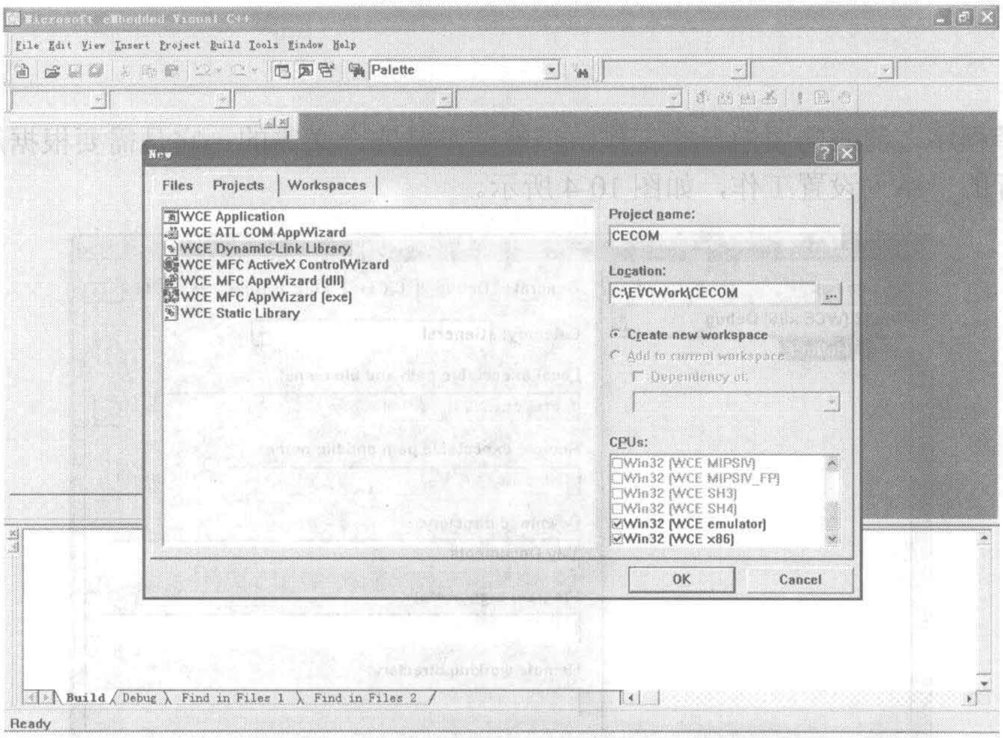


图 10-6

步骤 2 如图 10-7 所示，选择空的 DLL 工程。

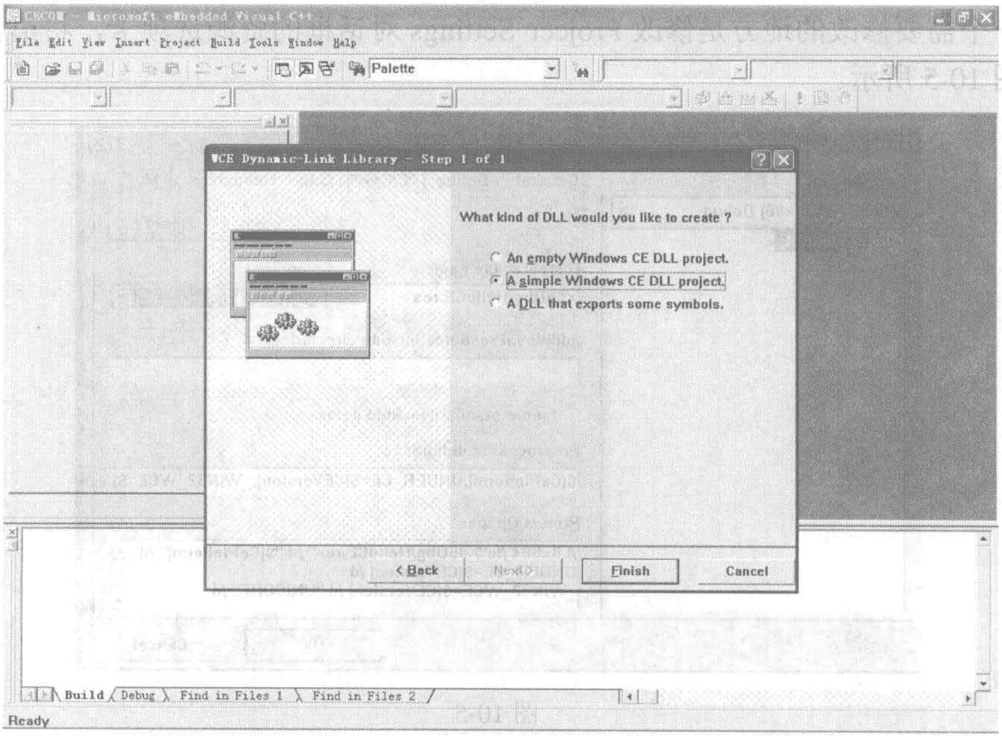


图 10-7

步骤 3 如图 10-8 所示，手工添加所需要的文件。

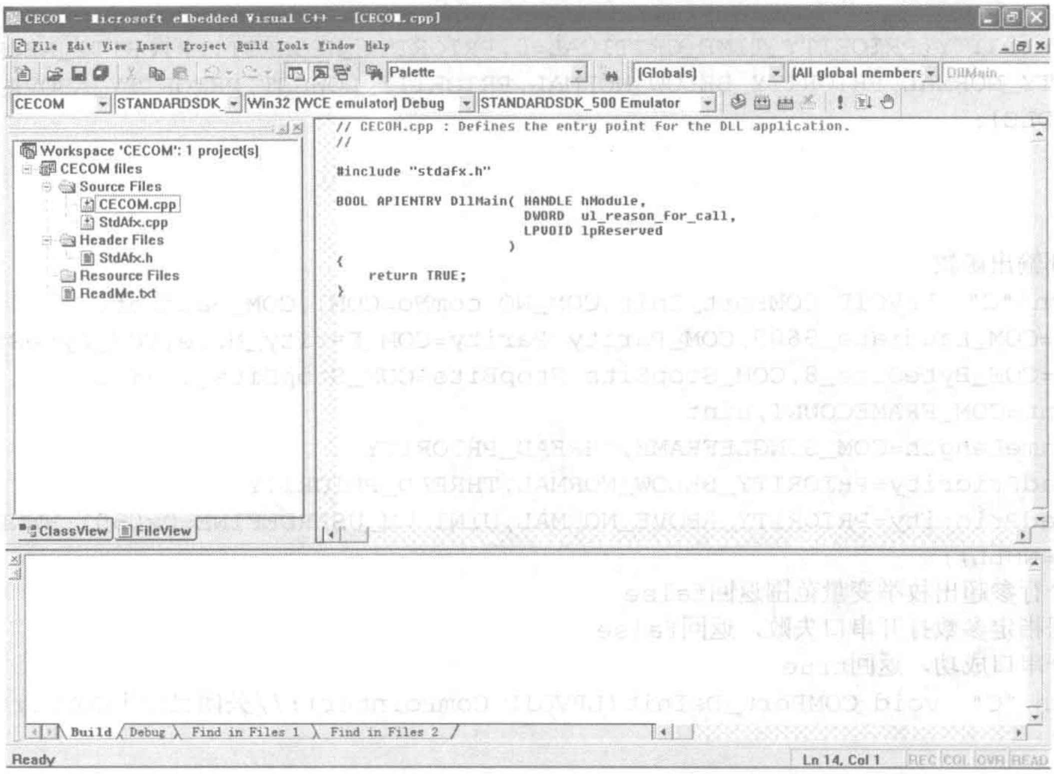


图 10-8

以下是串口通信程序所需要的头文件定义（需要手工添加）。

```
//定义串口需要的相关常量
//定义串口组件内部单帧接收发送数据最大长度
#define  COM_SINGLEFRAME 4096
//定义串口组件内部接收最大帧数
#define  COM_FRAMECOUNT 16

//串口定义
enum COM_NO{COM1=0,COM2,COM3,COM4,COM5,COM6,COM7,COM8,COM9};
//定义枚举
//串口停止位
enum COM_StopBits{COM_StopBits_1=0,COM_StopBits_1_5,COM_StopBits_2};
//串口传输字节长度
enum COM_ByteSize{COM_ByteSize_5=0, COM_ByteSize_6, COM_ByteSize_7,
COM_ByteSize_8};
//传输奇偶效验方式
enum COM_Parity{COM_Parity_None=0, COM_Parity_Odd, COM_Parity_Even,
COM_Parity_Mark, COM_Parity_Space};
//传输波特率
enum
COM_Baudrate{COM_Baudrate_110=0,COM_Baudrate_300,COM_Baudrate_600,COM_Baudrate
_1200,COM_Baudrate_2400,COM_Baudrate_4800,COM_Baudrate_9600,COM_Baudrate_19200
,COM_Baudrate_38400,COM_Baudrate_57600,COM_Baudrate_115200};
```

```
//定义优先级，我们特别定义是为了指导用户就用这几个
enum
THREAD_PRIORITY{PRIORITY_TIME_CRITICAL=0,PRIORITY_HIGHEST,PRIORITY_ABOVE_NORMA
L,PRIORITY_NORMAL,PRIORITY_BELOW_NORMAL,PRIORITY_LOWEST,PRIORITY_ABOVE_IDLE,PR
IORITY_IDLE};


//外部输出函数
extern "C"  LPVOID COMPort_Init(COM_NO comNo=COM1,COM_Baudrate
BaudRate=COM_Baudrate_9600,COM_Parity Parity=COM_Parity_None,COM_ByteSize
ByteSize=COM_ByteSize_8,COM_StopBits StopBits=COM_StopBits_1,uint
FrameCount=COM_FRAMECOUNT,uint
SingleFrameLength=COM_SINGLEFRAME,THREAD_PRIORITY
SendThreadPriority=PRIORITY_BELOW_NORMAL,THREAD_PRIORITY
RecvThreadPriority=PRIORITY_ABOVE_NORMAL,UINT WM_USERDEFINE=0x0501,HWND
MainHWnd=NULL);
//各个行参超出枚举变量范围返回false
//根据指定参数打开串口失败，返回false
//打开串口成功，返回true
extern "C"  void COMPort_DeInit(LPVOID ComPointer);//关闭已经用COMPort_Init()
打开的指针

//发送数据
extern "C"  void COMPort_SendByte(LPVOID ComPointer,byte* Data,UINT DataLen);
//Data为需要发送的数据数组首地址，DataLen为长度
//接收数据收到WM_USER以后被调用的处理函数
extern "C"  void COMPort_RecvByte(LPVOID ComPointer,byte* Data,UINT*
DataLen);
//Data为接收数组的首地址，DataLen为Data的长度，就是最大的返回个数，DataLen返回时候被
填实际个数
```

以下是串口通信程序的核心源码：

```
// COMPort.cpp : Defines the entry point for the DLL application.
//

#include "stdafx.h"
#include "COMPort.h"
#include "windows.h"
#include "VoidPointer.h"
#include "StreamBuffer.h"
#include "TCore.h"

#include "ceddk.h"


void COMPort_DeInitAll();//声明一下下
```

```
BOOL APIENTRY DllMain( HANDLE hModule,
                       DWORD  ul_reason_for_call,
                       LPVOID lpReserved
              )
{

        case DLL_PROCESS_DETACH:
          COMPort_DeInitAll();//即使用户没有显式调用Deinit，DLL退出时候自动调用
        break;
        case DLL_THREAD_ATTACH:
        break;
        case DLL_THREAD_DETACH:
        break;
        default:
        break;
    }

    return TRUE;
}

float ManProtect_count=0.1f;
void ManProtect()
{

}

//起作用的内部类对象
//内部串口类
class CCOMPort
{
private:
//串口名称，对应COMX对应"COMX:"，赋值在类实现中
 static WCHAR* COMNOSTRING[9];
 WCHAR* m_ComNoString;
 COM_NO m_ComNo;
//串口波特率
 static DWORD BAUDRATE[11];
 DWORD m_BaudRate;
//串口party
 static BYTE PARITY[5];
 BYTE m_Parity;
//串口字节尺寸
 static BYTE BYTESIZE[4];
 BYTE m_ByteSize;
//串口停止位
 static BYTE STOPBITS[3];
 BYTE m_StopBits;
```

```
//串口发送线程优先级，实际这个优先级赋给了线程发送缓冲区对象的内部线程的优先级指定
THREAD_PRIORITY m_SendThreadPriority;
//串口接收线程优先级，这个不是给线程接收缓冲区的内部线程优先级，这是接收串口事件的独立
//线程的优先级指定，线程接收缓冲区的内部线程优先级固定为PRIORITY_BELOW_NORMAL
THREAD_PRIORITY m_RecvThreadPriority;

//串口接收线程句柄
HANDLE m_RecvThread;

//打开串口
BOOL OpenCOM();

//关闭串口
void CloseCOM();

//配置串口PCB
BOOL ConfigDCB();

//发送缓冲区对象需要的内部缓冲空间指针
byte* SendBufferData;
//发送缓冲区对象

//接收缓冲区对象需要的内部缓冲空间指针
byte* RecvBufferData;

CStreamBuffer* SendBuffer;
//发送缓冲区静态回调，只要发送缓冲区有数据就由发送缓冲区对象内含的线程自动调用这个函数
static void SendCallBack(byte* Data,UINT DataLen,void* pUserClassPoint);

//串口发送需要的事件
OVERLAPPED SendOverLapped;
//接收缓冲区静态回调，只要接收缓冲区有数据就由接收缓冲区对象内含的线程自动调用这个函数
static void RecvCallBack(byte* Data,UINT DataLen,void* pUserClassPoint);
UINT m_WM_USERDEFINE;
HWND m_MainHWND;
byte* m_CurrentRecvBufferPoint;
UINT m_CurrentRecvBufferLen;

public:

static uint CECOMEVENT;

HANDLE m_hCom; //串口句柄


//串口接收线程退出信号灯，不需要了，因为使用WaitCommEvent(),关闭串口导致
WaitCommEvent()返回0
//导致接收线程从容退出
//HANDLE m_RecvThreadCloseHandle;
```

```
    //判断是否正确打开串口
    BOOL bOpened;

    //构造函数
    //CCOMPort(COM_NO comNo=COM1,COM_Baudrate
BaudRate=COM_Baudrate_9600,COM_Parity Parity=COM_Parity_None,COM_ByteSize
ByteSize=COM_ByteSize_8,COM_StopBits StopBits=COM_StopBits_1,uint
SingleFrameLength=COM_SINGLEFRAME,THREAD_PRIORITY
SendThreadPriority=PRIORITY_BELOW_NORMAL,THREAD_PRIORITY
RecvThreadPriority=PRIORITY_ABOVE_NORMAL,PFUserProcessCallBack
fRecvCallBack=NULL);
    CCOMPort(COM_NO comNo=COM1,COM_Baudrate
BaudRate=COM_Baudrate_9600,COM_Parity Parity=COM_Parity_None,COM_ByteSize
ByteSize=COM_ByteSize_8,COM_StopBits StopBits=COM_StopBits_1,uint
FrameCount=COM_FRAMECOUNT,uint
SingleFrameLength=COM_SINGLEFRAME,THREAD_PRIORITY
SendThreadPriority=PRIORITY_BELOW_NORMAL,THREAD_PRIORITY
RecvThreadPriority=PRIORITY_ABOVE_NORMAL,UINT WM_USERDEFINE=0x0501,HWND
MainHWND=NULL);

    //析构函数
    virtual ~CCOMPort();


    //让主函数调用的发送函数
    void ComSendData(byte* Data,UINT DataLen);
    //接收缓冲区对象
    CStreamBuffer* RecvBuffer;

    void ComRecvByte(byte* Data,UINT* DataLen);
    //Data为接收数组的首地址，DataLen为Data的长度，就是最大的返回个数，DataLen返回时候被
填实际个数

    //串口收发线程单帧长度
    uint m_SingleFrameLength;

    //串口收发线程最大帧数
    uint m_FrameCount;

    uint m_CECOMEVENT;
    };

    //给静态标签数据赋值
    WCHAR* CCOMPort::COMNOSTRING[] =
{L"COM1:",L"COM2:",L"COM3:",L"COM4:",L"COM5:",L"COM6:",L"COM7:",L"COM8:",L"COM
9:"};
    DWORD
CCOMPort::BAUDRATE[]={110,300,600,1200,2400,4800,9600,19200,38400,57600,115200
};
    BYTE  CCOMPort::PARITY[]={0,1,2,3,4};
```

```
BYTE  CCOMPort::BYTESIZE[]={5,6,7,8};
BYTE  CCOMPort::STOPBITS[]={0,1,2};
uint  CCOMPort::CECOMEVENT=EV_TXEMPTY|EV_RXCHAR|EV_CTS|EV_DSR|EV_RING;


//接收线程主体
//WINCE下的串口不接收异步事件，类似XP系统的overlapped结构是不能使用的，只有使用
WaitCommEvent
//这个东西在串口句柄无效的时候自动会产生返回0，用这个来控制线程
DWORD RecvThreadProc(LPVOID lpParam)
{
 CCOMPort* ComObject=(CCOMPort*)lpParam;
 DWORD dwCommModemStatus=0;
 byte* CurrentData=new
byte[ComObject->m_SingleFrameLength*ComObject->m_FrameCount];
 DWORD CurrentDataLen=0;
 BOOL bReadFileRet=true;
 while(true){
     //形式上的死循环
     WaitCommEvent(ComObject->m_hCom,&dwCommModemStatus,NULL);

     if (dwCommModemStatus==0){
         //ComObject关闭了串口，导致WaitCommEvent调用失败，我们从这里从容退出线程
         break;
     }
     else{
         //必须马上设置新中断条件
         SetCommMask(ComObject->m_hCom,ComObject->m_CECOMEVENT);

         //下面分别处理事件
         if ((dwCommModemStatus & EV_RXCHAR)!=0){

 bReadFileRet=ReadFile(ComObject->m_hCom,&CurrentData[0],ComObject->m_Singl
eFrameLength,&CurrentDataLen,NULL);
             if ((CurrentDataLen!=0) && (ComObject->RecvBuffer!=NULL)){
                 //将数据送走，直接送入接收缓冲区
                 //的判断，因为线程可能先起来

 ComObject->RecvBuffer->DataProcess(CurrentData,CurrentDataLen);
                 CurrentDataLen=0;
             }
         }
         if ((dwCommModemStatus & EV_TXEMPTY)!=0){
             //串口发送完毕一个数据
             ;//我们在发送端使用WriteFile一次发送，所以这里什么都不做
         }

         //其他我们都不干任何事情
     }
 }
```

```
  delete CurrentData;
  return 0;
  }


  ///////////////////////////发送线程和接收线程主体//////////////////////

  /////////////////////整个DLL管理的链表类////////////////////////////////
  CVoidpointerList COMList;
  ///////////////////////////外部输出函数//////////////////////////////////


  //BOOL COMPort_Init(COM_NO comNo,COM_Baudrate Baudrate,COM_Parity
Parity,COM_ByteSize ByteSize,COM_StopBits StopBits,uint
SingleFrameLength,THREAD_PRIORITY SendThreadPriority,THREAD_PRIORITY
RecvThreadPriority,PFUserProcessCallBack fRecvCallBack)
  LPVOID COMPort_Init(COM_NO comNo,COM_Baudrate Baudrate,COM_Parity
Parity,COM_ByteSize ByteSize,COM_StopBits StopBits,uint FrameCount,uint
SingleFrameLength,THREAD_PRIORITY SendThreadPriority,THREAD_PRIORITY
RecvThreadPriority,UINT WM_USERDEFINE,HWND MainHWND)
  {


   if ((comNo<COM1) || (comNo>COM9)){//我们明确规定枚举变量范围
       //传入的COM_NO不是我们规定的范围，是其他一个整数，其实是不在(0, 8)范围
       return NULL;
   }
   if ((Baudrate<COM_Baudrate_110) || (Baudrate>COM_Baudrate_115200)){
       return NULL;
   }
   if ((Parity<COM_Parity_None) || (Parity>COM_Parity_Space)){
       return NULL;
   }
   if ((ByteSize<COM_ByteSize_5) || (ByteSize>COM_ByteSize_8)){
       return NULL;
   }
   if ((StopBits<COM_StopBits_1) || (StopBits>COM_StopBits_2)){
       return NULL;
   }
   if ((SingleFrameLength<0) || (SingleFrameLength>COM_SINGLEFRAME)){
       return NULL;
   }
   if ((SendThreadPriority<PRIORITY_TIME_CRITICAL) ||
(SendThreadPriority>PRIORITY_IDLE)){
       return NULL;
   }
   if ((RecvThreadPriority<PRIORITY_TIME_CRITICAL) ||
(RecvThreadPriority>PRIORITY_IDLE)){
       return NULL;
   }
```

```
    CCOMPort* compointer=new
CCOMPort(comNo,Baudrate,Parity,ByteSize,StopBits,FrameCount,SingleFrameLength,
SendThreadPriority,RecvThreadPriority,WM_USERDEFINE,MainHWND);
    //MessageBox(NULL,_T("0"),_T(""),MB_OK);

    if (!compointer->bOpened){
        //打开串口失败
        delete compointer;
        compointer=NULL;
        return NULL;
    }
    //这里打开串口成功，将串口类句柄返回
    COMList.ADD((LPVOID)compointer);
    return compointer;//将本次申请成功的句柄返回调用函数
   }
   void COMPort_DeInit(LPVOID ComPointer)//关闭已经用COMPort_Init()打开的指针
   {
    if (ComPointer==NULL) return;
    LPVOID tmp=NULL;
    CCOMPort* compointer=NULL;
    for (int i=0;i<COMList.count;i++){
        tmp=COMList.GETVALUE(i);
        if ((tmp!=NULL)&&(tmp==ComPointer)){
            //找到这个串口实例
            compointer=(CCOMPort*)tmp;
            delete compointer;
            compointer=NULL;
            COMList.DELETENODE(i);
            break;
        }
    }

   }

   void COMPort_DeInitAll()
   {
    LPVOID tmp=NULL;
    CCOMPort* compointer=NULL;

    for (int i=0;i<COMList.count;i++){
        tmp=COMList.GETVALUE(i);
        if (tmp!=NULL){
            //找到这个串口实例
            compointer=(CCOMPort*)tmp;
            delete compointer;
            compointer=NULL;
        }
    }
```

```
  COMList.CLEAR();
 }
void COMPort_SendByte(LPVOID ComPointer,byte* Data,UINT DataLen)
{
 //发送数据
 if (ComPointer==NULL) return;
 LPVOID tmp=NULL;
 CCOMPort* compointer=NULL;
 for (int i=0;i<COMList.count;i++){
     tmp=COMList.GETVALUE(i);
     if ((tmp!=NULL)&&(tmp==ComPointer)){
        //找到这个串口实例
        compointer=(CCOMPort*)tmp;
        if ((compointer!=NULL) && (compointer->bOpened)){
           compointer->ComSendData(Data,DataLen);//将参数传入串口类相关函数
        }
        break;
     }
 }
}

void COMPort_RecvByte(LPVOID ComPointer,byte* Data,UINT* DataLen)
{
 //接收数据

 if (ComPointer==NULL) return;
 LPVOID tmp=NULL;
 CCOMPort* compointer=NULL;
 for (int i=0;i<COMList.count;i++){
     tmp=COMList.GETVALUE(i);
     if ((tmp!=NULL)&&(tmp==ComPointer)){
        //找到这个串口实例
        compointer=(CCOMPort*)tmp;
        if ((compointer!=NULL) && (compointer->bOpened)){
           compointer->ComRecvByte(Data,DataLen);
        }
        break;
     }
 }
}

////////////////////////////外部输出函数////////////////////////////////////

//内部串口类
//CCOMPort::CCOMPort(COM_NO comNo,COM_Baudrate Baudrate,COM_Parity
Parity,COM_ByteSize ByteSize,COM_StopBits StopBits,uint
SingleFrameLength,THREAD_PRIORITY SendThreadPriority,THREAD_PRIORITY
```

```
RecvThreadPriority,PFUserProcessCallBack fRecvCallBack)
  CCOMPort::CCOMPort(COM_NO comNo,COM_Baudrate Baudrate,COM_Parity
Parity,COM_ByteSize ByteSize,COM_StopBits StopBits,uint FrameCount,uint
SingleFrameLength,THREAD_PRIORITY SendThreadPriority,THREAD_PRIORITY
RecvThreadPriority,UINT WM_USERDEFINE,HWND MainHWND)
  {

   SendBufferData=NULL;
   RecvBufferData=NULL;
   RecvBuffer=NULL;
   SendBuffer=NULL;
   //通过外部包装 函数传入的时候，我们已经剔除了comNo<0或者comNo>8的情况
   //其实comNo<0或者comNo>8完全等同于comNo<COM1或者comNo>COM9，这里不再判断
   //传入的5个参数，都类似的被剔除了无效参数，所以能够在这里产生对象实体，所有参数均有效
   m_ComNo=comNo;
   m_ComNoString=COMNOSTRING[comNo];//得到"COMX:"
   m_BaudRate=BAUDRATE[Baudrate];//得到波特率
   m_Parity=PARITY[Parity];//得到Parity参数
   m_ByteSize=BYTESIZE[ByteSize];//得到ByteSize参数
   m_StopBits=STOPBITS[StopBits];//得到停止位
   m_FrameCount=FrameCount;//最大帧数
   m_SingleFrameLength=SingleFrameLength;//得到单帧最大长度
   m_SendThreadPriority=SendThreadPriority;//得到发送线程优先级
   m_RecvThreadPriority=RecvThreadPriority;//得到接收线程优先级
   m_WM_USERDEFINE=WM_USERDEFINE;//得到外部传来的消息ID定义
   m_MainHWND=MainHWND;//得到外部传来的接收消息的窗口句柄
   m_CECOMEVENT=CCOMPort::CECOMEVENT;//正常情况下应该干串口该干的事情,刚开始不做事情


   m_hCom=NULL;
   m_RecvThread=NULL;

   bOpened=false;//刚开始打开串口失败，必须加这句话，否则OpenCOM()影响流程
   //打开串口
   bOpened=OpenCOM();
   if (bOpened){

       //注意，线程调用有顺序，下面这些线程的启用顺序不能改变

       //产生传入两个缓冲类中的数据存储空间
       uint Len=m_FrameCount*m_SingleFrameLength;
       SendBufferData=new byte[Len];
       RecvBufferData=new byte[Len];


       //产生接收缓冲区
       RecvBuffer=new
CStreamBuffer(RecvBufferData,Len,PRIORITY_BELOW_NORMAL,RecvCallBack,this);
       //这里特别注意，接收线程优先级是外面送进来的，但是接收线程实际完成数据源的工作，处理
接收线程的
       //优先级我们设置为PRIORITY_BELOW_NORMAL，是因为这是接受缓冲区内部调度用户界面回
```

调函数的线程

```
    //优先级，一般接收线程优先级必须设置为比通常高一级，而接收缓冲优先级中的线程必须比普
通线程
    //还要低一级，仔细想一想这是为什么。

    //产生发送缓冲区
    SendBuffer=new
CStreamBuffer(SendBufferData,Len,m_SendThreadPriority,SendCallBack,this);
    //我们让发送缓冲区的内部调度线程和发送线程在同一个优先级
    //我们认为创建这个缓冲区和内部的线程永远不会发生错误

    //这里必须先创建缓冲区，因为实践证明，当接收线程一启动就可能发送数据，而这些数据是要
    //送入缓冲区的，如果这个时候缓冲区没有启动，系统出现大错误，指针错误
    //创建发送回调中需要的异步结构
    memset(&SendOverLapped,0,sizeof(OVERLAPPED));//数组清零
    SendOverLapped.hEvent=::CreateEvent(NULL,true,false,NULL);//create a
readevent and enable

    //创建收发线程，这个必须是打开成功才可以
      DWORD dwRecvThreadId;

  m_RecvThread=::CreateThread(NULL,0,(LPTHREAD_START_ROUTINE)RecvThreadProc,
this,0,&dwRecvThreadId);
    //这里的这个this，就正好传给了线程主体的唯一一个行参
    ::SetThreadPriority(m_RecvThread,(int)m_RecvThreadPriority);

  }
  //我们认为创建线程永远不会错误
  }

  CCOMPort::~CCOMPort()
  {
  CloseCOM();
  }

  BOOL CCOMPort::OpenCOM()
  {
  if (bOpened) return true;//串口已经打开，直接返回串口已经打开

  COMMTIMEOUTS CommTimeOuts;//声明超时结构
  //打开串口
  m_hCom=::CreateFile(m_ComNoString,GENERIC_READ|GENERIC_WRITE,//允许读写
                    0,//不共享
                      NULL,
                      OPEN_EXISTING,//已有设备
                      FILE_FLAG_OVERLAPPED,//选用异步方式
                      NULL);
  if (m_hCom==INVALID_HANDLE_VALUE){//返回INVALID_HANDLE_VALUE失败
      m_hCom=NULL;
```

```
        return false;
    }
    ::SetCommMask(m_hCom,CCOMPort::CECOMEVENT);//监视串口错误和事件信号改变和响铃三
种事件
    ::SetupComm(m_hCom,4096,4096);//设置串口输入输出缓冲区，这个值是微软标准4096
    //清空输入输出缓冲区
    ::PurgeComm(m_hCom,PURGE_TXABORT|PURGE_RXABORT|PURGE_TXCLEAR|PURGE_RXCLEAR
);

    CommTimeOuts.ReadIntervalTimeout=0xFFFFFFFF;
    CommTimeOuts.ReadTotalTimeoutMultiplier=0;
    CommTimeOuts.ReadTotalTimeoutConstant=0;
    CommTimeOuts.WriteTotalTimeoutMultiplier=10;
    CommTimeOuts.WriteTotalTimeoutConstant=1000;

       ::SetCommTimeouts(m_hCom,&CommTimeOuts);//设置读写超时设定

    //配置PCB
    return ConfigDCB();
   }

   BOOL CCOMPort::ConfigDCB()
   {
    if (m_hCom==NULL) return false;
    DCB dcb;
    ::GetCommState(m_hCom,&dcb);//得到默认串口参数设置
    dcb.fBinary=1;//设置为二进制数据模式
    dcb.fParity=1;//0;//奇偶校验不使能

    dcb.XonLim=500;
    dcb.XoffLim=500;
    dcb.XonChar=17;
    dcb.XoffChar=19;
    dcb.ErrorChar=0;


    dcb.BaudRate=m_BaudRate;
    dcb.ByteSize=m_ByteSize;
    dcb.Parity=m_Parity;
    dcb.StopBits=m_StopBits;

       ::SetCommState(m_hCom,&dcb);//设置串口参数
    return true;
   }

   void CCOMPort::CloseCOM()
   {
    if ((bOpened) && (m_hCom!=NULL)){
```

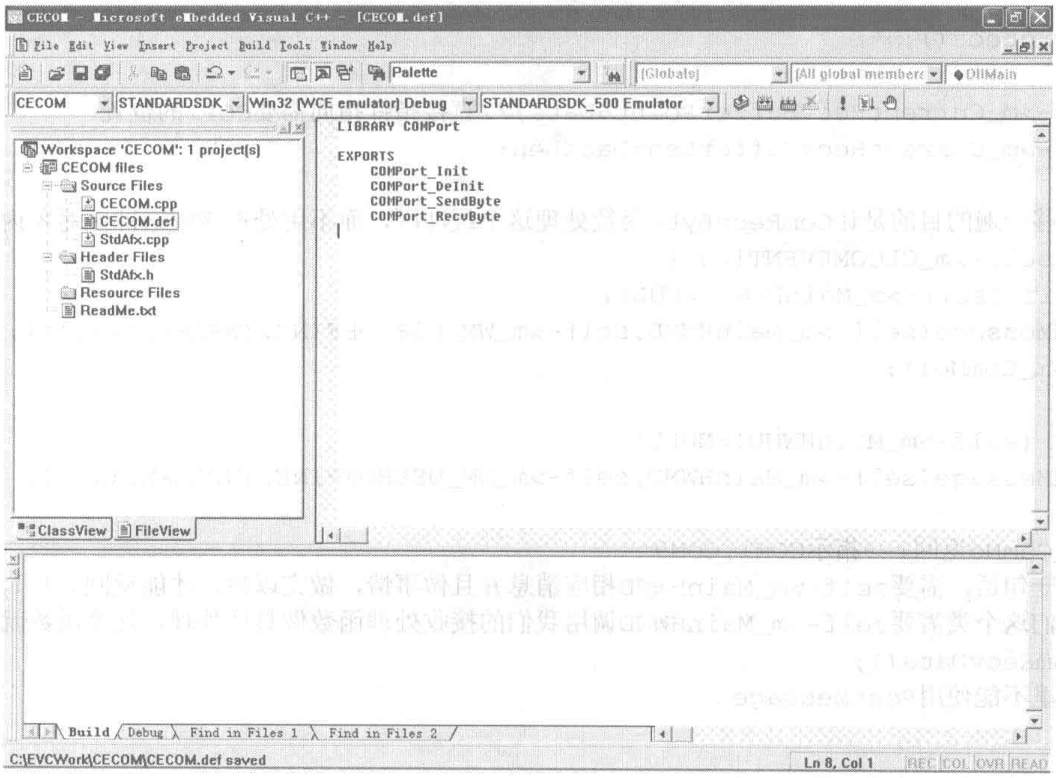
```
        //关闭所有的事件
        m_CECOMEVENT=0;
        ::CloseHandle(SendOverLapped.hEvent);
    SendOverLapped.hEvent=NULL;
        //必须关闭串口
        //清空输入输出缓冲区

    ::PurgeComm(m_hCom,PURGE_TXABORT|PURGE_RXABORT|PURGE_TXCLEAR|PURGE_RXCLEAR
);
        SetCommMask(m_hCom,m_CECOMEVENT);//关闭中断标签，以后线程再也不来数据了
        ::CloseHandle(m_hCom);
        m_hCom=NULL;
        //特别注意，这里关闭串口才导致接收线程中的WaitCommEvent()返回0，所以我们在这里等
接收线程结束
        WaitForSingleObject(m_RecvThread,INFINITE);
        //这以后，没有串口事件了

        //释放发送缓冲区对象，并且发送缓冲区对象的析构自动关闭发送缓冲区内含的线程
        delete SendBuffer;
        SendBuffer=NULL;
        //释放接收缓冲区对象，并且接收缓冲区对象的析构自动关闭接收缓冲区内含的线程
        delete RecvBuffer;
        RecvBuffer=NULL;

        delete SendBufferData;//删除缓冲区数据区
        SendBufferData=NULL;
        delete RecvBufferData;//删除缓冲区数据区
        RecvBufferData=NULL;

        bOpened=false;
    }
    }

    void CCOMPort::SendCallBack(byte* Data,UINT DataLen,void* pUserClassPoint)
    {
     //这个函数只被发送缓冲区类中的线程调用，而那个线程的优先级是比正常低一级
     //所以这个函数虽然定义在主线程函数中，但是，实际运行优先级比正常低一级
     CCOMPort* self=(CCOMPort*)pUserClassPoint;

     //发送缓冲区静态回调，只要发送缓冲区有数据就由发送缓冲区对象内含的线程自动调用这个函数
     //这里的Data和DataLen和ComSendData()传入的Data和DataLen不一样，这里是缓冲以后由缓
冲
     //内部线程调度以后出来的数据
     //将数据传到真正的当前发送缓冲区
     //发送事件
     int number=DataLen;
     int BeginPosition=0;//开始的位置
     DWORD writtenbyte=0;//开始被写的数据个数
     DWORD dwLastError;
     BOOL bWriteState;
```

```
    DWORD WaitForResult;

    do{

    bWriteState=WriteFile(self->m_hCom,&Data[BeginPosition],number,&writtenbyt
e,&self->SendOverLapped);
        if (!bWriteState){
            dwLastError=GetLastError();
            if (dwLastError==ERROR_INVALID_HANDLE) break;
            if (dwLastError==ERROR_IO_PENDING) break;
    WaitForResult=WaitForSingleObject(self->SendOverLapped.hEvent,INFINITE);
            if (WaitForResult==0){
                //本次发送完
                //if
(!GetOverlappedResult(ComObject->m_hCom,&SendOverLapped,&Beginwrittenbyte,true
)) break;
                //这个编译通不过，因为我们在PLATFORM BUILDER平台下编译的，WINCE没有异步
串口这样的用法
                //如果是XP平台应该打开这句话的屏蔽
                }
                else{
                    break;
                }
                number-=writtenbyte;
                BeginPosition+=writtenbyte;
            }
            else{
                number-=writtenbyte;
                BeginPosition+=writtenbyte;
            }
        }while(number>0);
        //发送事件
    }

void CCOMPort::ComSendData(byte* Data,UINT DataLen)
{
    //外部函数传入的发送函数，这是将发送数据送入发送缓冲区
    //这里我们只管送入发送缓冲区，发送缓冲区内的线程会自动调用发送缓冲区对象创建时候
    //送入的回调函数进行缓冲区输出处理
    if (SendBuffer==NULL) return;
    SendBuffer->DataProcess(Data,DataLen);
}

void CCOMPort::RecvCallBack(byte* Data,UINT DataLen,void* pUserClassPoint)
{
    //接收的缓冲出口，缓冲入口来自高优先级的接收线程
    //这个函数只被发送缓冲区类中的线程调用，而那个线程的优先级是比正常低一级
    //所以这个函数虽然定义在主线程函数中，但是，实际运行优先级比正常低一级
    CCOMPort* self=(CCOMPort*)pUserClassPoint;
```

```
ManProtect();

self->m_CurrentRecvBufferPoint=Data;//直接指针指向需要COPY的位置
self->m_CurrentRecvBufferLen=DataLen;

//转移一遍的目的是让ComRecvByte函数处理这个缓冲区，而不用处理接收缓冲区对象内部的东西
if (self->m_CECOMEVENT!=0){
    if (self->m_MainHWND!=NULL)
SendMessage(self->m_MainHWND,self->m_WM_USERDEFINE,(WPARAM)(self),(LPARAM)
(&self->m_ComNo));
}
//if (self->m_MainHWND!=NULL)
SendMessage(self->m_MainHWND,self->m_WM_USERDEFINE,(WPARAM)(&self->m_ComNo
),0);
//m_ComNo返回1-9指示COM1,COM9
//这一句话，需要self->m_MainHWND相应消息并且做事情，做完以后，才能返回！！！
//我们这个类需要self->m_MainHWND调用我们的接收处理函数做具体处理，这个函数就是
//ComRecvData();
//这里不能使用PostMessage


}

void CCOMPort::ComRecvByte(byte* Data,UINT* DataLen)
{

//self->m_MainHWND响应自定义消息，必须调用的函数，目的是填数据出去
//Data是外部C#传入的接收首址，DataLen传入是Data长度，出去的时候是实际
    if (m_CurrentRecvBufferLen>=*DataLen){
    m_CurrentRecvBufferLen=*DataLen;
}

//wchar_t c[100];
//_ultow(m_CurrentRecvBufferLen,c,10);
//MessageBox(NULL,c,_T("ComRecvByte"),MB_OK);

memcpy(Data,m_CurrentRecvBufferPoint,m_CurrentRecvBufferLen);
*DataLen=m_CurrentRecvBufferLen;//把实际传输出去的数据长度也传出去
}
```

最后，如图 10-9 所示，需要手工添加 CECOM.def 文件，将 DLL 中外部接口函数导出。文件内容如下：

```
LIBRARY COMPort
EXPORTS
 COMPort_Init
 COMPort_DeInit
 COMPort_SendByte
 COMPort_RecvByte
```



CECOM - Microsoft eMbedded Visual C++ - [CECOM.def]
File Edit View Insert Project Build Tools Window Help
Palette
[Globals]
[All global members]
DllMain
CECOM
STANDARDSDK
Win32 (WCE emulator) Debug
STANDARDSDK_500 Emulator
Workspace 'CECOM': 1 project(s)
CECOM files
Source Files
CECOM.cpp
CECOM.def
StdAfx.cpp
Header Files
StdAfx.h
Resource Files
ReadMe.txt
ClassView
FileView
LIBRARY COMPort
EXPORTS
COMPort_Init
COMPort_DeInit
COMPort_SendByte
COMPort_RecvByte
Build
Debug
Find in Files 1
Find in Files 2
C:\EVCWork\CECOM\CECOM.def saved
Ln 8, Col 1
REC
COL
OVR
READ

图 10-9

第 11 章　.NET 应用程序开发

11.1　.NET2.0 精简版基本控件使用方法

如图 11-1 所示，根据如下向导建立智能设备应用程序工程，本例程将介绍.NET2.0 精简版主要控件的使用方法。

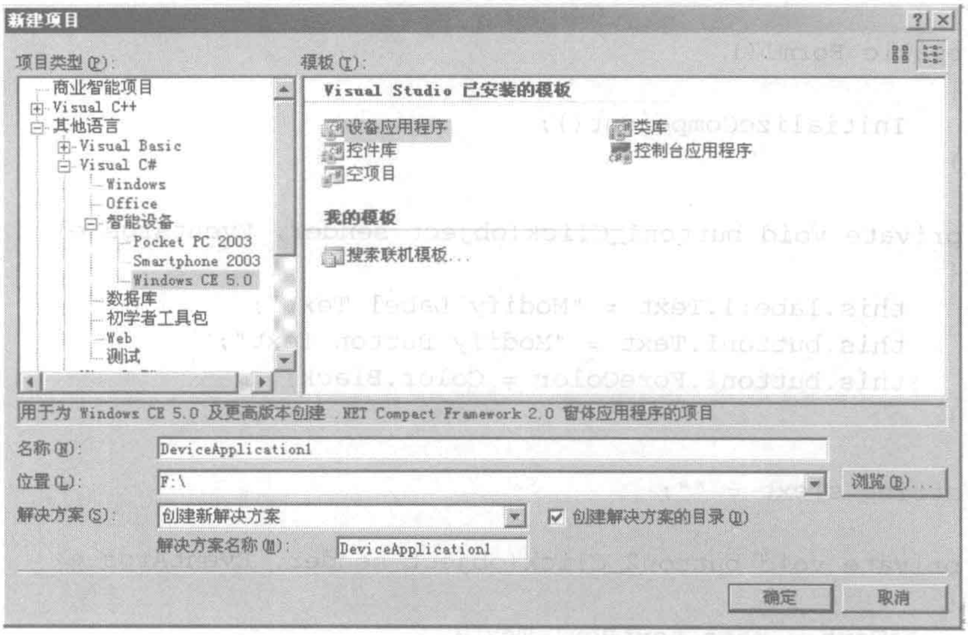


图 11-1

新建一个 Form，如图 11-2 所示添加控件。

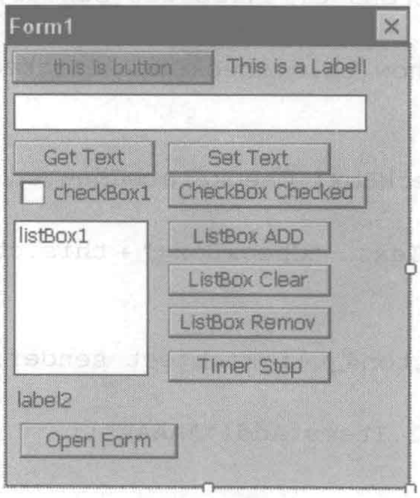


图 11-2

Form1.cs 文件源码内容如下：

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;

namespace DeviceApplication1
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }

        private void button1_Click(object sender, EventArgs e)
        {
            this.label1.Text = "Modify Label Text";
            this.button1.Text = "Modify Button Text";
            this.button1.ForeColor = Color.Black;
        }

        string sText = "";

        private void button2_Click(object sender, EventArgs e)
        {
            sText = this.textBox1.Text;
            MessageBox.Show(sText);
        }

        private void button3_Click(object sender, EventArgs e)
        {
            MessageBox.Show(this.checkBox1.Checked.ToString());
        }

        private void checkBox1_CheckStateChanged(object sender, EventArgs e)
        {
            this.label1.Text = "CheckBox=" + this.checkBox1.Checked.ToString();
        }

        private void button4_Click(object sender, EventArgs e)
        {
            this.listBox1.Items.Add("AAAAA");
        }

        private void button5_Click(object sender, EventArgs e)
        {
```

```
        this.listBox1.Items.Clear();
    }

    private void button6_Click(object sender, EventArgs e)
    {
        if (this.listBox1.Items.Count > 0)
        {
            this.listBox1.Items.RemoveAt(0);
        }
        else
        {
            MessageBox.Show("No Data!");
        }
    }


    bool b_Timer = true;
    long iTimerCount = 0;

    private void timer1_Tick(object sender, EventArgs e)
    {
        iTimerCount++;
        if (this.b_Timer)
        {
            if (iTimerCount % 10 == 0)
            {
                this.label2.Text = System.DateTime.Now.ToString();
            }
        }
    }

    private void button7_Click(object sender, EventArgs e)
    {
        this.b_Timer = false;
    }

    private void button8_Click(object sender, EventArgs e)
    {
        this.textBox1.Text = " Set Text";
    }

    Form2 f2 = null;
    private void button9_Click(object sender, EventArgs e)
    {
        if (f2 == null)
        {
            f2 = new Form2();
            f2.ShowDialog();
        }
        f2 = null;
```

```
        }
    }
}
```

建立第二个 Form，如图 11-3 所示添加控件。

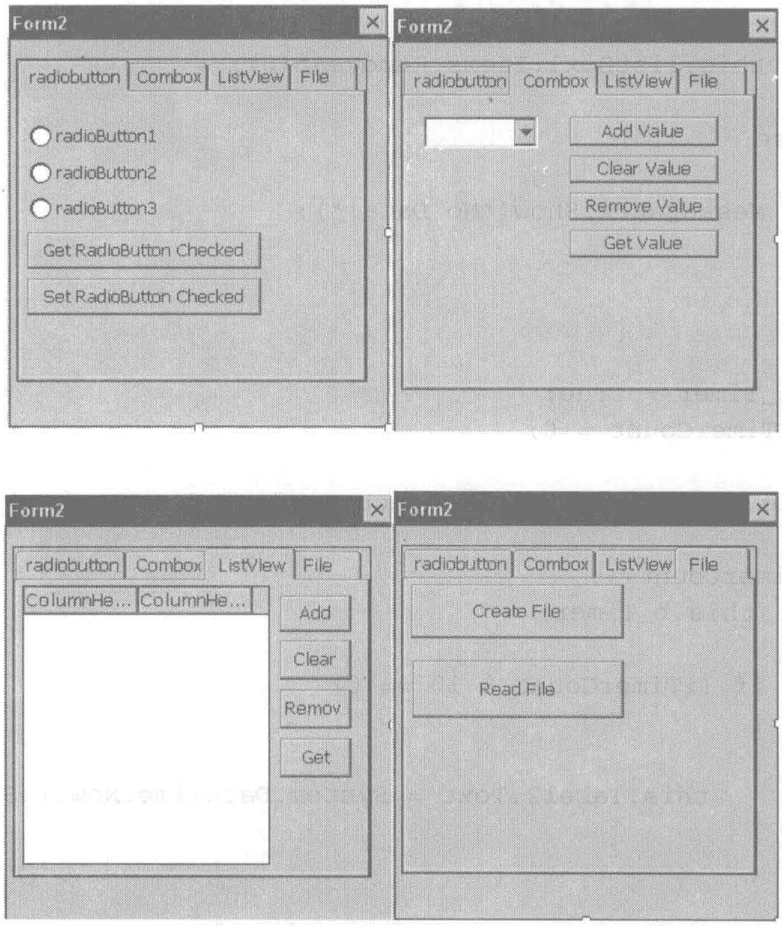


图 11-3

Form2.cs 源代码文件内容如下所示：

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;
using System.IO;

namespace DeviceApplication1
{
    public partial class Form2 : Form
    {
        public Form2()
```

```
        {
            InitializeComponent();
        }

        private void button1_Click(object sender, EventArgs e)
        {
            MessageBox.Show("R1="+this.radioButton1.Checked.ToString());
            MessageBox.Show("R2=" + this.radioButton2.Checked.ToString());
            MessageBox.Show("R3=" + this.radioButton3.Checked.ToString());
        }

        private void button2_Click(object sender, EventArgs e)
        {
            this.radioButton1.Checked = true;
        }

        private void radioButton1_CheckedChanged(object sender, EventArgs e)
        {
            this.label1.Text =
this.radioButton1.Text+"="+this.radioButton1.Checked.ToString();
        }

        private void radioButton2_CheckedChanged(object sender, EventArgs e)
        {
            this.label1.Text = this.radioButton2.Text + "=" +
this.radioButton2.Checked.ToString();
        }

        private void radioButton3_CheckedChanged(object sender, EventArgs e)
        {
            this.label1.Text = this.radioButton3.Text + "=" +
this.radioButton3.Checked.ToString();
        }

        private void button3_Click(object sender, EventArgs e)
        {
            this.comboBox1.Items.Add("AAA");
        }

        private void button4_Click(object sender, EventArgs e)
        {
            this.comboBox1.Items.Clear();
        }

        private void button5_Click(object sender, EventArgs e)
        {
            if (this.comboBox1.Items.Count > 0)
            {
                this.comboBox1.Items.RemoveAt(0);
            }
        }
```

```
        private void button6_Click(object sender, EventArgs e)
        {
            MessageBox.Show(this.comboBox1.SelectedItem.ToString());
        }

        private void button7_Click(object sender, EventArgs e)
        {
            ListViewItem item=new ListViewItem("C1");
            this.listView1.Items.Add(item);

            this.listView1.Items[this.listView1.Items.Count -
1].SubItems.Add("C2");
        }

        private void button8_Click(object sender, EventArgs e)
        {
            this.listView1.Items.Clear();
        }

        private void button9_Click(object sender, EventArgs e)
        {
            if (this.listView1.Items.Count > 0)
            {
                this.listView1.Items.RemoveAt(0);
            }
        }

        private void button10_Click(object sender, EventArgs e)
        {
            MessageBox.Show(this.listView1.Items[4].Text);
            MessageBox.Show(this.listView1.Items[4].SubItems[0].Text);
        }

        private void listView1_SelectedIndexChanged(object sender, EventArgs
e)
        {
            MessageBox.Show("Change!");
        }

        private void button11_Click(object sender, EventArgs e)
        {
            System.IO.StreamWriter sw = new StreamWriter("a.txt");
            for (int i = 0; i < 100; i++)
            {
                sw.WriteLine("Hello");
            }
            sw.Close();

        }

        private void button12_Click(object sender, EventArgs e)
        {
```

```
            System.IO.StreamReader sr = new StreamReader("a.txt");
            while (!sr.EndOfStream)
            {
                MessageBox.Show(sr.ReadLine());
            }
            sr.Close();
        }
    }
}
```

11.2 C#开发串口通信例程

如图 11-4 所示，根据向导，建立智能设备类库项目，本例程将利用托管 API 代码实现串口通信程序类库。

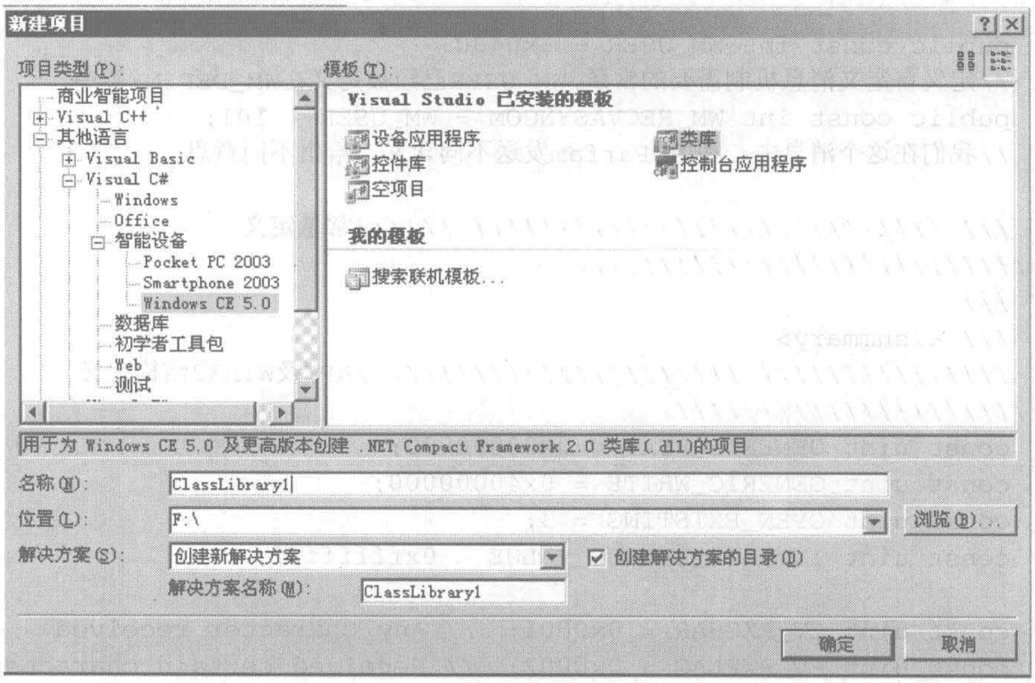


图 11-4

项目 ClassLibrary1 的 Class1.cs 的源代码内容如下：

```
using System;
using Microsoft.WindowsCE.Forms;
using System.Runtime.InteropServices;
using System.Windows.Forms;
using System.Threading;


namespace CeAsynComDemo
{
    /// <summary>
    /// CeAsynCom 的摘要说明。
    /// </summary>
```

```
    public class CeAsynCom
    {
        //读线程向主窗体发送了有进入数据的消息，主窗体调用这个回调函数进行一次将所有数据处理完成的功能
        public delegate void SingleDeal_RecvData(byte[] DataBuffer, int DataBufferLength);//让外面函数调用这个回调函数

        /// <summary>
        /// ////////////////////////////////////////常量定义////////////////////////////////////////
        //定义串口需要的枚举变量
        public enum COM_StopBits { _1, _1_5, _2 };//停止位
        public enum COM_ByteSize { _5, _6, _7, _8 };//传输字节长度
        public enum COM_Parity { None, Odd, Even, Mark, Space };//奇偶效验方式
        public enum COM_Baudrate { _110, _300, _600, _1200, _2400, _4800, _9600, _19200, _38400, _57600, _115200 };//串口波特率

        public const int WM_USER = 0x0400;
        //定义自定义消息机制需要的常量,WM_USER已经被定义在WM_DEFINE类中
        public const int WM_RECVASYNCOM = WM_USER + 101;
        //我们在这个消息中，利用wParfam发送不同含义，给出不同意思

        /// ////////////////////////////////////////常量定义////////////////////////////////////////
        ///
        /// </summary>
        ////////////////////////////////////////API及Win32结构定义////////////////////////////////
        const uint GENERIC_READ = 0x80000000;
        const uint GENERIC_WRITE = 0x40000000;
        const uint OPEN_EXISTING = 3;
        const uint INVALID_HANDLE_VALUE = 0xffffffff;

        const uint EV_RXCHAR = 0x0001;  // Any Character received
        const uint EV_RXFLAG = 0x0002;  // Received certain character
        const uint EV_TXEMPTY = 0x0004;  // Transmitt Queue Empty
        const uint EV_CTS = 0x0008;  // CTS changed state
        const uint EV_DSR = 0x0010;  // DSR changed state
        const uint EV_RLSD = 0x0020;  // RLSD changed state
        const uint EV_BREAK = 0x0040;  // BREAK received
        const uint EV_ERR = 0x0080;  // Line status error occurred
        const uint EV_RING = 0x0100;  // Ring signal detected

        const uint CECOMEVENT = EV_TXEMPTY | EV_RXCHAR | EV_CTS | EV_DSR | EV_RING;

        const uint PURGE_TXABORT = 0x0001;
        const uint PURGE_RXABORT = 0x0002;
        const uint PURGE_TXCLEAR = 0x0004;
        const uint PURGE_RXCLEAR = 0x0008;


        ///
```

```
        ///声明控制串口需要的超时结构
        [StructLayout(LayoutKind.Sequential)]
        internal class COMMTIMEOUTS
        {
           public uint ReadIntervalTimeout;
           public uint ReadTotalTimeoutMultiplier;
           public uint ReadTotalTimeoutConstant;
           public uint WriteTotalTimeoutMultiplier;
           public uint WriteTotalTimeoutConstant;
        }
        //得到串口超时结构函数
        //BOOL GetCommTimeouts(HANDLE hFile,LPCOMMTIMEOUTS lpCommTimeouts);
        [DllImport("Coredll.dll", CallingConvention =
CallingConvention.Winapi)]
        internal static extern bool GetCommTimeouts(
           IntPtr hFile,
           COMMTIMEOUTS CommTimeouts
        );
        //注意，这里有一个非常有意思的事情，在MSDN中，这个函数在Serdev.lib中连接
        //但是，函数本身不存在Serdev.dll，而是Coredll.dll

        //设置串口超时结构
        //BOOL SetCommTimeouts(HANDLE hFile,LPCOMMTIMEOUTS lpCommTimeouts);
        [DllImport("Coredll.dll", CallingConvention =
CallingConvention.Winapi)]
        internal static extern bool SetCommTimeouts(
           IntPtr hFile,
           COMMTIMEOUTS CommTimeouts
        );


        [DllImport("Coredll.dll", CallingConvention =
CallingConvention.Winapi)]
        internal static extern IntPtr CreateFile(
           string lpFileName,
           uint dwDesiredAccess,
           uint dwShareMode,
           IntPtr lpSecurityAttributes,
           uint dwCreationDisposition,
           uint dwFlagsAndAttributes,
           IntPtr hTemplateFile
        );

        //关闭串口句柄
        /*BOOL CloseHandle(HANDLE hObject);*/
        [DllImport("Coredll.dll", CallingConvention =
CallingConvention.Winapi)]
        internal static extern bool CloseHandle(IntPtr hObject);
        //SetCommMask()
        [DllImport("Coredll.dll", CallingConvention =
```

```
CallingConvention.Winapi)]
        internal static extern int SetCommMask(IntPtr hFile, uint dwEvtMask);

        //设置串口缓冲区
        //BOOL SetupComm(HANDLE hFile,DWORD dwInQueue,DWORD dwOutQueue);
        [DllImport("Coredll.dll", CallingConvention =
CallingConvention.Winapi)]
        internal static extern bool SetupComm(
            IntPtr hFile,
            int dwInQueue,
            int dwOutQueue
        );

        //PurgeComm()定义
        [DllImport("Coredll.dll", CallingConvention =
CallingConvention.Winapi)]
        internal static extern bool PurgeComm(
            IntPtr hFile,
            uint dwFlags
        );

        [StructLayout(LayoutKind.Sequential)]//按照顺序布局
        public class DCB
        {
            public uint DCBlength;
            public uint BaudRate;
            public int Flags;
            public ushort wReserved;
            public ushort XonLim;
            public ushort XoffLim;
            public byte ByteSize;
            public byte Parity;
            public byte StopBits;
            public byte XonChar;
            public byte XoffChar;
            public byte ErrorChar;
            public byte EofChar;
            public byte EvtChar;
            public ushort wReserved1;
        }

        //11、得到串口的DCB参数
        /*BOOL GetCommState(HANDLE hFile,LPDCB lpDCB);*/
        [DllImport("Coredll.dll", CallingConvention =
CallingConvention.Winapi)]
        internal static extern int GetCommState(
            IntPtr hFile,
            DCB dcb
        );

        //12、设置串口的DCB参数
```

```
        /*BOOL SetCommState(HANDLE hFile,LPDCB lpDCB);*/
        [DllImport("Coredll.dll", CallingConvention =
CallingConvention.Winapi)]
        internal static extern int SetCommState(
            IntPtr hFile,
            DCB dcb
        );

        //串口写出函数

        [DllImport("Coredll.dll", CallingConvention =
CallingConvention.Winapi)]
        internal static extern bool WriteFile(
            IntPtr hFile,
            byte[] lpBuffer,
            uint nNumberOfBytesToWrite,
            ref uint lpNumberOfBytesWritten,
            IntPtr lpOverlapped
        );

        //串口等待事件，阻塞事件，非常类似的中断响应函数
        [DllImport("Coredll.dll", EntryPoint = "WaitCommEvent")]
        public static extern int WaitCommEvent(
            IntPtr hFile,
            ref int lpEvtMask,
            IntPtr lpOverlapped
        );

        //串口读取数据
        [DllImport("Coredll.dll", CallingConvention =
CallingConvention.Winapi)]
        internal static extern bool ReadFile(
            IntPtr hFile,
            byte[] lpBuffer,
            uint nNumberOfBytesToRead,
            ref uint lpNumberOfBytesRead,
            IntPtr lpOverlapped
        );

        ///////////////////////////////////////////API及Win32结构定义
/////////////////////////////

        bool bDisposed;//专门用于析构函数的标签
        MessageWindow pMessageWindow;//指向串口接收到数据以后，发送给的主窗口，由于
WINCE窗口没有句柄，所以给MessageWindow代理窗口
        string mSelectCom;//串口名称，必须类似"COM1","COM2"
        COM_Baudrate mBaudRate;//波特率
        COM_Parity mParity;//奇偶校验
        COM_ByteSize mByteSize;//数据位长度
        COM_StopBits mStopBits;//停止位
```

```
        //指明当前串口是否已经打开
        bool bOpenCom;
        IntPtr mCommHandle;//准备打开的串口句柄
        //一个类似中断处理机制的四循环线程，必须强制关断

        //管理串口发、收事件的线程
        RWThread mRWThread = null;//产生线程的普通类，提供Execute()作为托管的被线程
作为Execute()执行的实体
        Thread RWThreadObject = null;//真正的线程实体
        //判断是否有发送队列的线程
        Control_WThread mControl_WThread = null;
        Thread Control_WThreadObject = null;
        bool bCurrentSending = false;
        int iCurrentSendingCount = 0;

        public const uint BUFFERLENGTH = 4096;//单帧长度
        public const uint BUFFERAMOUNT = 16;//缓冲区一共多少帧
        public const int BUFFERREMAIND = 10;//发送接收超过尾部不能超过
X-BUFFERREMAIND

        byte[][] ReadBuffer = new byte[BUFFERAMOUNT][];
        byte[][] WriteBuffer = new byte[BUFFERAMOUNT][];
        uint[] ReadBuffer_FrameCount = new uint[BUFFERAMOUNT];//每一帧数据中实
际个数
        int[] WriteBuffer_FrameCount = new int[BUFFERAMOUNT];//每一帧数据中实际
个数
        //前后指针
        uint ReadBuffer_q = 0;
        uint ReadBuffer_h = 0;
        uint WriteBuffer_q = 0;
        uint WriteBuffer_h = 0;


        //循环指针,静态方法
        internal static uint P_INC(uint pointLocation)
        {
            uint p;
            p = pointLocation;
            p++;
            if (p >= BUFFERAMOUNT)
            {//头指针小于最大值加一
                p = 0;
            }
            return (p);
        }

        internal static uint P_DEC(uint pointLocation)
        {
            uint p;
            p = pointLocation;
            if (p <= 0)
```

```
        {//如果头指针回到起始位置就退回到循环队列起始位置前一个位置即最大标号减一的位
置
            p = BUFFERAMOUNT - 1;
        }
        else
        {
            p--;
        }
        return (p);
    }

    public CeAsynCom(MessageWindow pMessageWindow, string SelectCom,
COM_Baudrate BaudRate, COM_Parity Parity, COM_ByteSize ByteSize, COM_StopBits
StopBits)
    {

        //首先分配缓冲区内存
        for (int i = 0; i < BUFFERAMOUNT; i++)
        {
            ReadBuffer[i] = new byte[BUFFERLENGTH];
            WriteBuffer[i] = new byte[BUFFERLENGTH];
        }
        //这样可以通过Array.Copy(数组名,...);长度也分配妥当

        //强调一下
        WriteBuffer_q = 0;
        WriteBuffer_h = 0;
        ReadBuffer_q = 0;
        ReadBuffer_h = 0;
        //
        // TODO: 在此处添加构造函数逻辑
        //
        this.pMessageWindow = pMessageWindow;
        mSelectCom = SelectCom;//串口名称，必须类似"COM1:","COM2:", 这里特别注
意和XP不同，不是COM1，COM2，而是COM1:，COM2:
        mBaudRate = BaudRate;//波特率
        mParity = Parity;//奇偶校验
        mByteSize = ByteSize;//数据位长度
        mStopBits = StopBits;//停止位
        bOpenCom = false;//初始生成对象都没有打开串口

        COMMTIMEOUTS commtimeouts = new COMMTIMEOUTS();
        //打开串口
        mSelectCom = "COM7:";
        this.mCommHandle = CreateFile(mSelectCom,
                                      GENERIC_READ | GENERIC_WRITE,//可以读写
                                      0,//not shared?
                                      IntPtr.Zero,//
                                      OPEN_EXISTING,
```

```
                                        0,//WINCE不支持Overlapped, 这里只能是
                                        IntPtr.Zero);
            if ((uint)mCommHandle == INVALID_HANDLE_VALUE)
            {
                throw new Exception("串口打开失败");
            }
            else
            {//串口打开成功
                bOpenCom = true;//打开串口成功
                //MessageBox.Show("串口打开成功");
                //设置同步需要WaitCommEvent监视的几种事件
                SetCommMask(this.mCommHandle, CECOMEVENT);//监视的多种事件
                //设置串口基本缓冲区
                if (SetupComm(this.mCommHandle, 4096, 4096) == false)
                {
                    throw new Exception("设置串口缓冲区不成功");
                }
                else
                {
                    PurgeComm(this.mCommHandle, PURGE_TXABORT | PURGE_RXABORT |
PURGE_TXCLEAR | PURGE_RXCLEAR);

                    commtimeouts.ReadIntervalTimeout = uint.MaxValue;
                    commtimeouts.ReadTotalTimeoutMultiplier = 0;
                    commtimeouts.ReadTotalTimeoutConstant = 0;
                    commtimeouts.WriteTotalTimeoutMultiplier = 10;
                    commtimeouts.WriteTotalTimeoutConstant = 1000;

                    SetCommTimeouts(this.mCommHandle, commtimeouts);
                    //超时设置结束

                    //根据传入设置串口参数
                    SetOpenCommstate(mBaudRate, mParity, mByteSize, mStopBits);
                    this.CreateThreads();
                }

            }

            //专门用于析构函数的标签的设置
            bDisposed = false;
        }

        ~CeAsynCom()
        {//这个函数永远不能主动调用，它必然在对象失去生存期的时候被系统调用
            //路线A
            Dispose(false);//整个类析构，仅仅清理CLR不能的工作，因为CLR能够清理的工作，
在走到
            //这里的时候已经完成
        }
```

```
public void Dispose()
{//这个函数可以被程序员在不需要对象的时候，直接调用对象.Dispose()方法
   //来马上释放对象
   //路线B
   if (!bDisposed)
   {
      //路线C
      Dispose(true);//不是析构产生的动作，可以被主动调用，CLR能够做的和不能
够做的都得做
      GC.SuppressFinalize(this);
      bDisposed = true;
   }
}

protected void Dispose(bool bDisposing)
{
   if (bDisposing)
   {
      //对受控制资源进行清理，这里一般没有代码
      ;//路线D
   }
   //路线E
   //对不受CLR控制的资源进行清理
   if (this.bOpenCom)
   {
      this.bOpenCom = false;
      //取消线程
      if (this.RWThreadObject != null)
      {
         this.mRWThread.Dispose();
         this.mRWThread = null;
         this.RWThreadObject = null;
      }
      if (this.Control_WThreadObject != null)
      {
         this.mControl_WThread.Dispose();
         this.mControl_WThread = null;
         this.Control_WThreadObject = null;
      }

      //在构造函数中取得串口句柄
      CloseHandle(mCommHandle);
      //我们由于大量使用new，没有回收，所以我们在这里强迫回收，效果出奇的好
      GC.Collect();
      GC.WaitForPendingFinalizers();

   }
}
```

```
    bool MessageByPass = false;
    public void ManyDeal_RecvData(SingleDeal_RecvData method)
    {
        //我们多次将数据利用回调模式送到外部进行处理
        byte[] CurrentData = new byte[CeAsynCom.BUFFERLENGTH];
        int CurrentDataLength;

        if (MessageByPass == false)
        {
            MessageByPass = true;
            //如果没有阻塞，说明可以处理
            if (ReadBuffer_h == ReadBuffer_q)
            {
                this.ReadBuffer_h = 0;
                this.ReadBuffer_q = 0;
                MessageByPass = false;
                return;
            }

            while (ReadBuffer_h != ReadBuffer_q)
            {//多次调用
                //单次处理
                CurrentDataLength =
(int)ReadBuffer_FrameCount[ReadBuffer_h];
                Array.Copy(ReadBuffer[ReadBuffer_h], 0, CurrentData, 0,
CurrentDataLength);
                method(CurrentData, CurrentDataLength);
                ReadBuffer_h = CeAsynCom.P_INC(ReadBuffer_h);
            }

            this.ReadBuffer_h = 0;
            this.ReadBuffer_q = 0;
            MessageByPass = false;
        }
    }

    ////////////////////////////////////
    /////得到串口相关参数
    public void GetOpenCommstate(DCB dcb)
    {
        if (bOpenCom)
        {
            GetCommState(mCommHandle, dcb);

            //这里已经得到了dcb,我们同时将这个数据传递给本类的m族变量

            //处理波特率
            switch (dcb.BaudRate)
            {
                case 110:
```

```
        mBaudRate = COM_Baudrate._110;
        break;
    case 300:
        mBaudRate = COM_Baudrate._300;
        break;
    case 600:
        mBaudRate = COM_Baudrate._600;
        break;
    case 1200:
        mBaudRate = COM_Baudrate._1200;
        break;
    case 2400:
        mBaudRate = COM_Baudrate._2400;
        break;
    case 4800:
        mBaudRate = COM_Baudrate._4800;
        break;
    case 9600:
        mBaudRate = COM_Baudrate._9600;
        break;
    case 19200:
        mBaudRate = COM_Baudrate._19200;
        break;
    case 38400:
        mBaudRate = COM_Baudrate._38400;
        break;
    case 57600:
        mBaudRate = COM_Baudrate._57600;
        break;
    case 115200:
        mBaudRate = COM_Baudrate._115200;
        break;
    default:
        mBaudRate = COM_Baudrate._9600;
        break;
}

//处理奇偶效验方式
switch (dcb.Parity)
{
    case 0:
        mParity = COM_Parity.None;
        break;
    case 1:
        mParity = COM_Parity.Odd;
        break;
    case 2:
        mParity = COM_Parity.Even;
        break;
    case 3:
        mParity = COM_Parity.Mark;
        break;
```

```
            case 4:
                mParity = COM_Parity.Space;
                break;
            default:
                mParity = COM_Parity.None;
                break;
        }

        //处理传输字节长度
        switch (dcb.ByteSize)
        {
            case 5:
                mByteSize = COM_ByteSize._5;
                break;
            case 6:
                mByteSize = COM_ByteSize._6;
                break;
            case 7:
                mByteSize = COM_ByteSize._7;
                break;
            case 8:
                mByteSize = COM_ByteSize._8;
                break;
            default:
                mByteSize = COM_ByteSize._8;
                break;
        }

        //处理停止位
        switch (dcb.StopBits)
        {
            case 0:
                mStopBits = COM_StopBits._1;
                break;
            case 1:
                mStopBits = COM_StopBits._1_5;
                break;
            case 2:
                mStopBits = COM_StopBits._2;
                break;
            default:
                mStopBits = COM_StopBits._1;
                break;
        }
    }
}

void CreateThreads()
```

```
        {

            //产生线程
            this.mRWThread = new RWThread(this);
            //委托方式制定线程实际执行函数
            RWThreadObject = new Thread(new
ThreadStart(this.mRWThread.Execute));
            //按照线程类中的执行函数，我们开始线程
            RWThreadObject.Priority = ThreadPriority.AboveNormal;
            RWThreadObject.Start();

            this.mControl_WThread = new Control_WThread(this);
            this.Control_WThreadObject = new Thread(new
ThreadStart(this.mControl_WThread.Execute));
            this.Control_WThreadObject.Priority = ThreadPriority.BelowNormal;
            this.Control_WThreadObject.Start();
        }

        /////设置串口相关参数
        public void SetOpenCommstate(COM_Baudrate BaudRate, COM_Parity Parity,
COM_ByteSize ByteSize, COM_StopBits StopBit)
        {
            DCB dcb = new DCB();
            if (bOpenCom)
            {


                GetCommState(mCommHandle, dcb);

                //下面是为了全收全发#0-#255，必须完成的规定,是不变的
                dcb.Flags = 1;
                //为了进行二进制OR运算
                //praityCheck=true
                dcb.Flags = dcb.Flags | 2;
                //DtrControl=DtrEnable
                dcb.Flags = dcb.Flags | 0x10;
                //RtsControl = RtsEnable
                dcb.Flags = dcb.Flags | 0x1000;

                dcb.XonLim = 500;
                dcb.XoffLim = 500;
                dcb.XonChar = 17;
                dcb.XoffChar = 19;
                dcb.ErrorChar = 0;
                //以下是修改部分
                //必须首先得到参数


                //处理波特率
                switch (mBaudRate)
                {
```

```
        case COM_Baudrate._110:
            dcb.BaudRate = 110;
            break;
        case COM_Baudrate._300:
            dcb.BaudRate = 300;
            break;
        case COM_Baudrate._600:
            dcb.BaudRate = 600;
            break;
        case COM_Baudrate._1200:
            dcb.BaudRate = 1200;
            break;
        case COM_Baudrate._2400:
            dcb.BaudRate = 2400;
            break;
        case COM_Baudrate._4800:
            dcb.BaudRate = 4800;
            break;
        case COM_Baudrate._9600:
            dcb.BaudRate = 9600;
            break;
        case COM_Baudrate._19200:
            dcb.BaudRate = 19200;
            break;
        case COM_Baudrate._38400:
            dcb.BaudRate = 38400;
            break;
        case COM_Baudrate._57600:
            dcb.BaudRate = 57600;
            break;
        case COM_Baudrate._115200:
            dcb.BaudRate = 115200;
            break;
        default:
            dcb.BaudRate = 9600;
            break;
    }

    //处理奇偶效验方式
    switch (mParity)
    {
        case COM_Parity.None:
            dcb.Parity = 0;
            break;
        case COM_Parity.Odd:
            dcb.Parity = 1;
            break;
        case COM_Parity.Even:
            dcb.Parity = 2;
            break;
        case COM_Parity.Mark:
            dcb.Parity = 3;
```

```
                break;
            case COM_Parity.Space:
                dcb.Parity = 4;
                break;
            default:
                dcb.Parity = 0;
                break;
        }

        //处理传输字节长度
        switch (mByteSize)
        {
            case COM_ByteSize._5:
                dcb.ByteSize = 5;
                break;
            case COM_ByteSize._6:
                dcb.ByteSize = 6;
                break;
            case COM_ByteSize._7:
                dcb.ByteSize = 7;
                break;
            case COM_ByteSize._8:
                dcb.ByteSize = 8;
                break;
            default:
                dcb.ByteSize = 8;
                break;
        }

        //处理停止位
        switch (mStopBits)
        {
            case COM_StopBits._1:
                dcb.StopBits = 0;
                break;
            case COM_StopBits._1_5:
                dcb.StopBits = 1;
                break;
            case COM_StopBits._2:
                dcb.StopBits = 2;
                break;
            default:
                dcb.StopBits = 0;
                break;
        }

        //写入
        SetCommState(mCommHandle, dcb);
    }
}
```

```
        public void ComSendData(byte[] InData, int WriteBufferLength)
        {

            int realLength = WriteBufferLength >= (int)(CeAsynCom.BUFFERLENGTH)
- CeAsynCom.BUFFERREMAIND ? (int)(CeAsynCom.BUFFERLENGTH) -
CeAsynCom.BUFFERREMAIND : WriteBufferLength;
            Array.Copy(InData, 0, WriteBuffer[WriteBuffer_q], 0, realLength);
            WriteBuffer_FrameCount[WriteBuffer_q] = WriteBufferLength;
            WriteBuffer_q = P_INC(WriteBuffer_q);
            if (WriteBuffer_q == WriteBuffer_h)
            {
                WriteBuffer_q = P_DEC(WriteBuffer_q);
            }
        }



        //////////////////////////////////////线程类
///////////////////////////////////////
        internal class RWThread
        {
            bool bDisposed;//专门用于析构函数的标签

            CeAsynCom mParent;
            public RWThread(CeAsynCom mParent)
            {
                this.mParent = mParent;
                this.bDisposed = false;
            }

            ~RWThread()
            {
                //路线A
                Dispose(false);//整个类析构，仅仅清理CLR不能的工作，因为CLR能够清理的
工作，在走到
                //这里的时候已经完成
            }
            public void Dispose()
            {//这个函数可以被程序员在不需要对象的时候，直接调用对象.Dispose()方法
                //来马上释放对象
                //路线B
                if (!bDisposed)
                {
                    //路线C
                    Dispose(true);//不是析构产生的动作，可以被主动调用，CLR能够做的和
不能够做的都得做
                    GC.SuppressFinalize(this);
                    bDisposed = true;
                }
            }
            protected void Dispose(bool bDisposing)
            {
```

```
            if (bDisposing)
            {
                //对受控制资源进行清理，这里一般没有代码
                ;//路线D
            }
            //路线E
            //对不受CLR控制的资源进行清理
        }


        public void Execute()
        {
            int dwCommModemStatus = 0;
            uint lpNumberOfBytesWritten = 0;
            byte[] Current1Byte = new byte[1];
            byte[] CurrentRecvBytes = new byte[CeAsynCom.BUFFERLENGTH];
            uint lpNumberOfBytesRead = 0;


            //线程主要函数
            while (this.mParent.mRWThread != null)
            {
                //死循环
                WaitCommEvent(this.mParent.mCommHandle, ref
dwCommModemStatus, IntPtr.Zero);//阻塞函数，只有事件发生
                SetCommMask(this.mParent.mCommHandle, CECOMEVENT);//监视的
多种事件
                ////////处理事件
                if ((dwCommModemStatus & EV_RXCHAR) != 0)
                {
                    ReadFile(this.mParent.mCommHandle, CurrentRecvBytes,
CeAsynCom.BUFFERLENGTH, ref lpNumberOfBytesRead, IntPtr.Zero);

                    if (lpNumberOfBytesRead != 0)
                    {
                        //将Buffer送走
                        if (lpNumberOfBytesRead >= CeAsynCom.BUFFERLENGTH -
CeAsynCom.BUFFERREMAIND)
                        {
                            lpNumberOfBytesRead = CeAsynCom.BUFFERLENGTH -
CeAsynCom.BUFFERREMAIND;
                        }
                        Array.Copy(CurrentRecvBytes, 0,
this.mParent.ReadBuffer[this.mParent.ReadBuffer_q], 0,
(int)lpNumberOfBytesRead);
```

```
this.mParent.ReadBuffer_FrameCount[this.mParent.ReadBuffer_q] =
lpNumberOfBytesRead;
                                this.mParent.ReadBuffer_q =
CeAsynCom.P_INC(this.mParent.ReadBuffer_q);
                                if (mParent.ReadBuffer_q == mParent.ReadBuffer_h)
                                {
                                    mParent.ReadBuffer_q =
CeAsynCom.P_DEC(mParent.ReadBuffer_q);
                                }
                                Message msg =
Message.Create(this.mParent.pMessageWindow.Hwnd, CeAsynCom.WM_RECVASYNCOM,
(IntPtr)0, (IntPtr)0);
                                MessageWindow.PostMessage(ref msg);
                            }
                        }
                        if ((dwCommModemStatus & EV_TXEMPTY) != 0)
                        {
                            if (this.mParent.iCurrentSendingCount <
this.mParent.WriteBuffer_FrameCount[this.mParent.WriteBuffer_h])
                            {
                                Current1Byte[0] =
this.mParent.WriteBuffer[this.mParent.WriteBuffer_h][this.mParent.iCurrentSend
ingCount++];
                                WriteFile(this.mParent.mCommHandle, Current1Byte, 1,
ref lpNumberOfBytesWritten, IntPtr.Zero);
                            }
                            else
                            {
                                this.mParent.bCurrentSending = false;
                                this.mParent.WriteBuffer_h =
CeAsynCom.P_INC(this.mParent.WriteBuffer_h);
                                if (this.mParent.WriteBuffer_q ==
this.mParent.WriteBuffer_h)
                                {
                                    this.mParent.WriteBuffer_h = 0;
                                    this.mParent.WriteBuffer_q = 0;
                                }
                            }
                        }
                        /////////处理事件

                    }//while... end
                }

            }


        /////监管是否有发送的线程类
        internal class Control_WThread
```

```
    {
        bool bDisposed;//专门用于析构函数的标签

        CeAsynCom mParent;
        public Control_WThread(CeAsynCom mParent)
        {
            this.mParent = mParent;
            this.bDisposed = false;
        }

        ~Control_WThread()
        {
            //路线A
            Dispose(false);//整个类析构，仅仅清理CLR不能的工作，因为CLR能够清理的
工作，在走到
            //这里的时候已经完成
        }
        public void Dispose()
        {//这个函数可以被程序员在不需要对象的时候，直接调用对象.Dispose()方法
            //来马上释放对象
            //路线B
            if (!bDisposed)
            {
                //路线C
                Dispose(true);//不是析构产生的动作，可以被主动调用，CLR能够做的和
不能够做的都得做
                GC.SuppressFinalize(this);
                bDisposed = true;
            }
        }
        protected void Dispose(bool bDisposing)
        {
            if (bDisposing)
            {
                //对受控制资源进行清理，这里一般没有代码
                ;//路线D
            }

            //路线E
            //对不受CLR控制的资源进行清理
        }



        public void Execute()
        {

            uint lpNumberOfBytesWritten = 0;
            byte[] Current1Byte = new byte[1];

            while (this.mParent.mControl_WThread != null)
```

```
                {
                    if (this.mParent.bCurrentSending == false)
                    {
                        if (this.mParent.WriteBuffer_h !=
this.mParent.WriteBuffer_q)
                        {
                            this.mParent.iCurrentSendingCount = 0;
                            //开始发送第一个字节
                            this.mParent.bCurrentSending = true;//在发送中断开判断
                            //启动第一个发送
                            Current1Byte[0] =
this.mParent.WriteBuffer[this.mParent.WriteBuffer_h][this.mParent.iCurrentSend
ingCount++];
                            WriteFile(this.mParent.mCommHandle, Current1Byte, 1,
ref lpNumberOfBytesWritten, IntPtr.Zero);
                            //这件事情一定会引起RWThreadObject线程中的EV_TXEMPTY事
件。在那里接着发
                        }
                    }

                }//while... end
            }

        }

        /////////////////////////////////////线程类
//////////////////////////////////////////


    }

}
```

第 12 章　WINCE 6 数据库开发

在 WINCE6 下竟然没有 SQLServerCE 数据库存在了。SQLServer CE 数据库历经多个版本，除 Pocket Access 之外，这几乎是 WINCE 上关系数据库的唯一选择。事实上，Oracle 与 Sybase 都有对应 WINCE 的版本，但极少有人使用。WINCE6 应该对应的是 SQLServer2005 的精简版本，但由于种种原因，WINCE6 没有将它作为标准组件包含在内。本章将介绍如何手工添加"Microsoft SQL Server Compact Edition"这一关系数据库到 WINCE6 之中，并以源码方式介了数据库应用程序开发方法。

12.1　Windows CE 6 的数据库版本

在微软的帮助文档里说：SQLCE2.0 以及 SQLCEMobile 已经被删除了。在帮助文件：ms-help://MS.VSCC.v80/MS.VSIPCC.v80/MS.WindowsCE.v60.en/CE_BuildSystem/html/c47b46de-c4dd-4b98-9385-943c3c9a8d22.htm 中找到 Removed Sysgen Variables. 的位置：ms-help://MS.VSCC.v80/MS.VSIPCC.v80/MS.WindowsCE.v60.en/CE_BuildSystem/html/55a6495d-cda2-458c-abb5-08d3e31347b3.htm

其中注明了以下 3 个环境变量都被删除了：

SYSGEN_SQL2000_DP

SYSGEN_SQLCE20

SYSGEN_SQLMOBILE

实际上，经过测试，SYSGEN_SQLCE20，SYSGEN_SQLMOBILE 这两个环境变量还有效，CE2.0 还可以正常运行，不过.NET2.0 不支持它了，CE6 也不再使用.NET1.1 进行开发了，等于没用。

SYSGEN_SQLMOBILE 也有用，但是 SQLCEMobile 的注册表文件没有，只能添加 DLL，也无法实现。

事实上，微软已经放弃了 SQLCE2.0（也叫 SqlServerCE2000）以及 SQLCEMobile，目前微软官方网站提供的 Microsoft SQL Server Compact Edition，其实就是 SQLServerCE2005，有时也称为 SqlServer 2005 精简版。

微软官方网站提供的 SQLServerCE2005 的"部署"方案，是叫我们将 CAB 包手动安装到 WINCE 当中，这只能应用于手持式永不掉电的设备。真正有用的解决方案是将 SQLServerCE2005 数据库打包到 NK.bin 当中。

以下是详细解决方案：

首先在一个没有 SQLServerCE2005 的 NK 上安装一个 CAB 包 sqlce30.wce5.x86。文件位于 C:\Program Files\Microsoft Visual Studio 8\SmartDevices\SDK\SQL Server\Mobile\v3.0\wce500\x86 下，有这一个就可以正常操作数据库了。

如图 12-1 所示，利用注册表查看器，查找添加了哪些注册表项，再手工抄写出来。

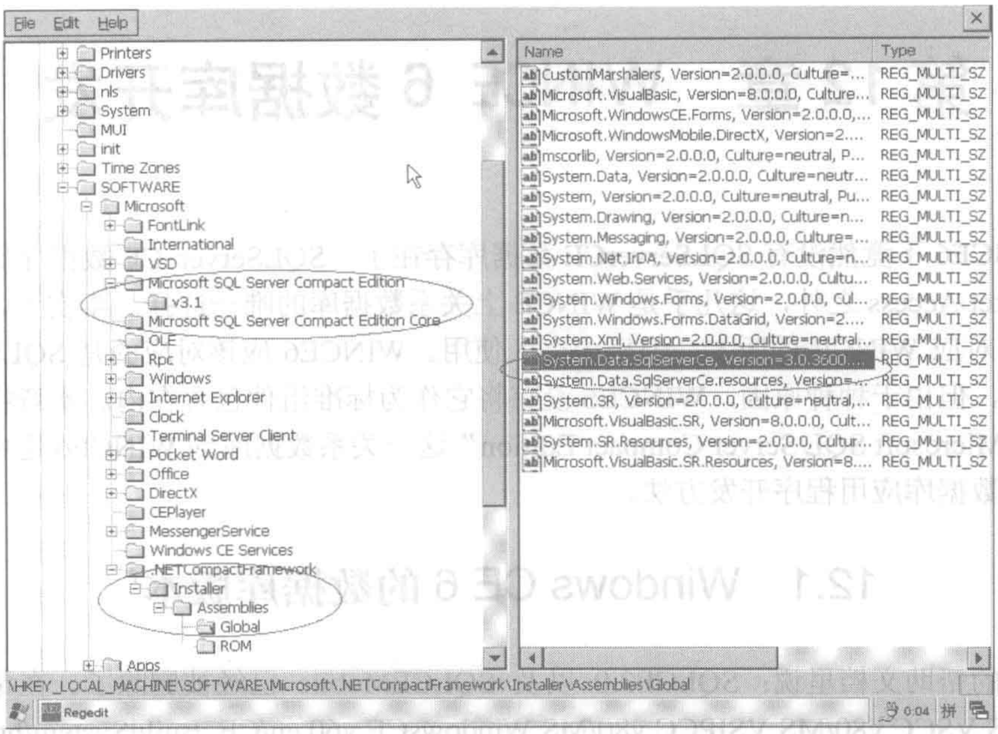


图 12-1

到 Windows 目录下查找安装了哪些文件，并复制到硬盘上，备用。

文件列表如下：

GAC_System.Data.SqlServerCe.resources_v3_0_3600_0_czh-CHS_1.dll

GAC_System.Data.SqlServerCe_v3_0_3600_0_cneutral_1.dll

sqlceer30CN.dll

sqlceme30.dll

sqlceqp30.dll

sqlcese30.dll

最后，我们需要将这些文件在 Project.bib 中指定复制到 NK.bin 当中，如图 12-2 所示。

修改后的文件如下所示（注意路径不要写错了）：

```
;
; Copyright (c) Microsoft Corporation.  All rights reserved.
;
;
; Use of this sample source code is subject to the terms of the Microsoft
; license agreement under which you licensed this sample source code. If
; you did not accept the terms of the license agreement, you are not
; authorized to use this sample source code. For the terms of the license,
; please see the license agreement between you and Microsoft or, if applicable,
; see the LICENSE.RTF on your install media or the root of your tools
installation.
; THE SAMPLE SOURCE CODE IS PROVIDED "AS IS", WITH NO WARRANTIES.
;
MODULES
```

```
; Name             Path                                          Memory Type
; ---------------  ---------------------------------------------
-----------

FILES
; Name             Path                                          Memory Type
; ---------------  ---------------------------------------------
-----------
GAC_System.Data.SqlServerCe.resources_v3_0_3600_0_czh-CHS_1.dll
D:\WINCE600\OSDesigns\SqlCE3\GAC_System.Data.SqlServerCe.resources_v3_0_3600_0
_czh-CHS_1.dll nk
GAC_System.Data.SqlServerCe_v3_0_3600_0_cneutral_1.dll
D:\WINCE600\OSDesigns\SqlCE3\GAC_System.Data.SqlServerCe_v3_0_3600_0_cneutral_
1.dll nk
sqlceer30CN.dll   D:\WINCE600\OSDesigns\SqlCE3\sqlceer30CN.dll nk
sqlceme30.dll     D:\WINCE600\OSDesigns\SqlCE3\sqlceme30.dll nk
sqlceqp30.dll     D:\WINCE600\OSDesigns\SqlCE3\sqlceqp30.dll nk
sqlcese30.dll     D:\WINCE600\OSDesigns\SqlCE3\sqlcese30.dll nk
```

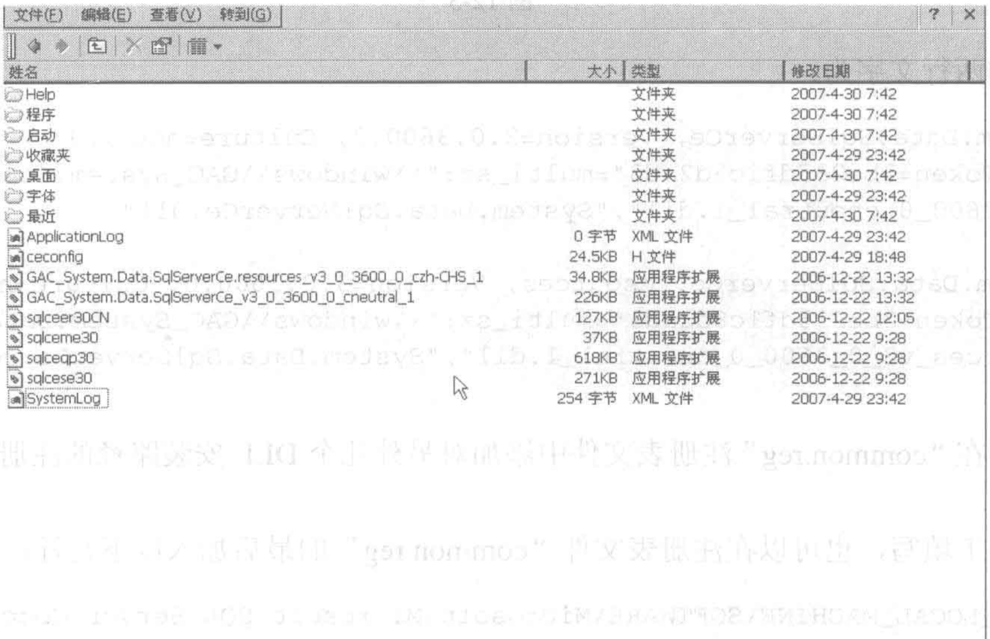


图 12-2

手工添加注册表项：

首先需要在 netcfv2.reg 文件中添加对 System.Data.SqlServerCe.dll 的引用。

修改后的文件见本目录下的“netcfv2.reg”文件

修改位置如图 12-3 所示。

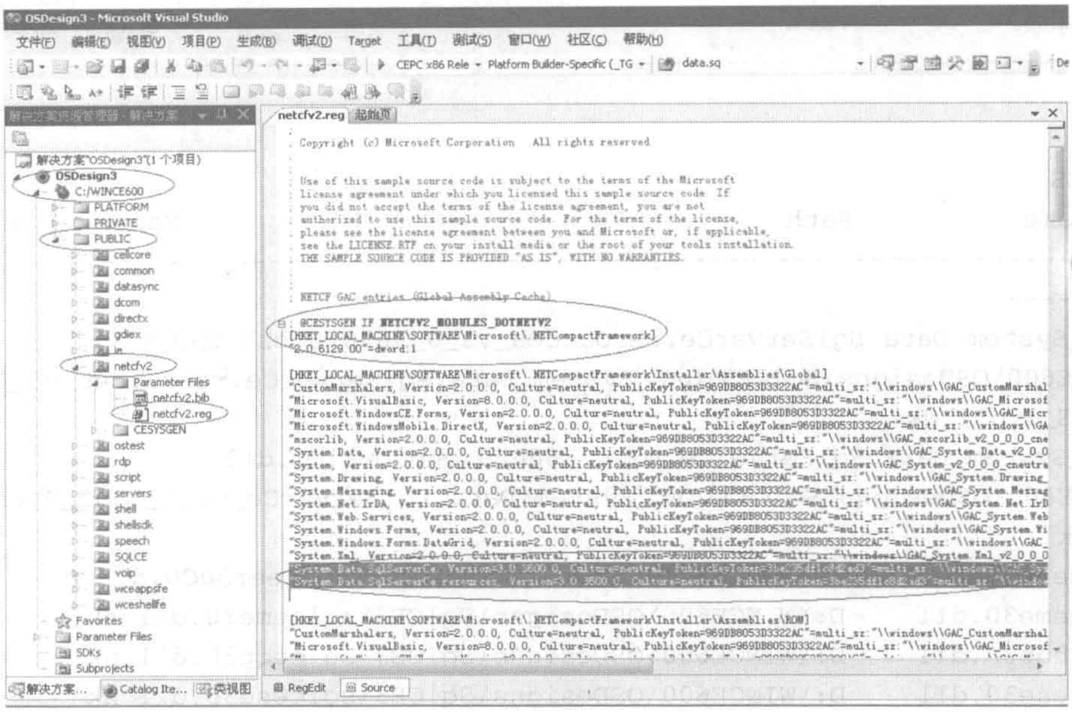


图 12-3

添加的两行文字为：

```
"System.Data.SqlServerCe, Version=3.0.3600.0, Culture=neutral, PublicKeyToken=3be235df1c8d2ad3"=multi_sz:"\\windows\\GAC_System.Data.SqlServerCe_v3_0_3600_0_cneutral_1.dll","System.Data.SqlServerCe.dll"

"System.Data.SqlServerCe.resources, Version=3.0.3600.0, Culture=neutral, PublicKeyToken=3be235df1c8d2ad3"=multi_sz:"\\windows\\GAC_System.Data.SqlServerCe.resources_v3_0_3600_0_cneutral_1.dll","System.Data.SqlServerCe.resources.dll"
```

其次要在“common.reg”注册表文件中添加对另外几个 DLL 安装路径的注册表项，如图 12-4 所示。

可以手工填写，也可以在注册表文件“common.reg”的最后加入以下几行：

```
[HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Microsoft SQL Server Compact Edition\v3.1]
"Nativedir"="\\windows"
[HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Microsoft SQL Server Compact Edition\v3.1]
"Installdir"="\\windows"
[HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Microsoft SQL Server Compact Edition Core]
"BldNum"=dword:e10
[HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Microsoft SQL Server Compact Edition Core]
"MinorVersion"=dword:0
[HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Microsoft SQL Server Compact Edition Core]
"MajorVersion"=dword:3
```

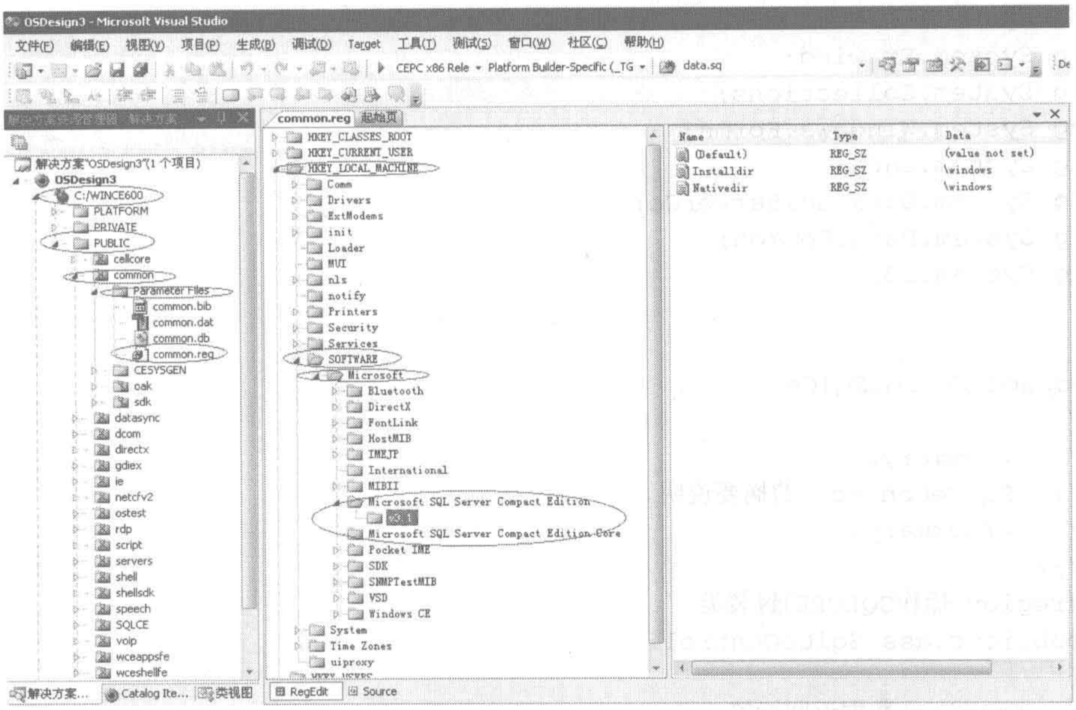


图 12-4

按照以上步骤进行操作，就可以在 WINCE 6 中实现 SQLServer2005 精简版数据库的操作。

12.2　.NET 操作 SqlServerCE 的数据库类

如图 12-5 所示，根据向导建立 SqlCeControl 智能设备类库项目。

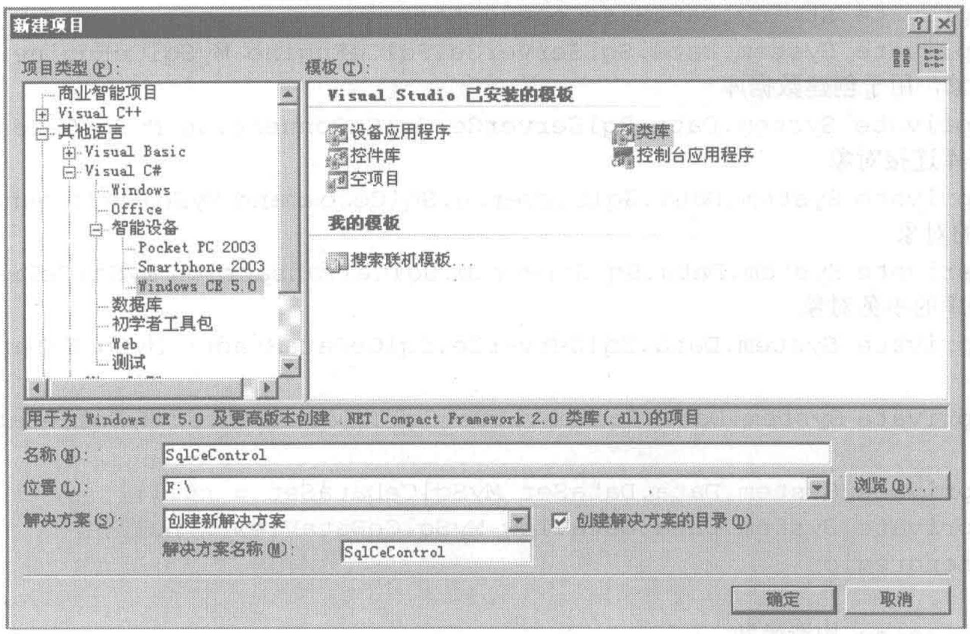


图 12-5

以下是 SqlCeControl.cs 源代码文件的内容：

```
using System;
using System.Drawing;
using System.Collections;
using System.Windows.Forms;
using System.Data;
using System.Data.SqlServerCe;
using System.Data.Common;
using System.IO;


namespace CotrolSqlCe
{
    /// <summary>
    /// SqlCeControl 的摘要说明。
    /// </summary>
    ///
    #region 操作SQLCE的封装类
    public class SqlCeControl
    {
        #region 数据返回标签
        public const int OP_SUCCESS = 0;
        public const int OP_SYSTEMERR = -1;
        public const int OP_DATABASECONNECTIONERR = -10;
        public const int OP_DATABASEEXISTS = 1;
        public const int OP_DATABASENOEXISTS = 2;
        public const int OP_ROLLBACK = 3;
        #endregion

        #region 类成员变量
        private string DataBaseName = "";//数据库的名字
        private string DataBaseConn = "";
        private System.Data.SqlServerCe.SqlCeEngine MySqlCeEngine = null;//
数据库引擎对象，用于创建数据库
        private System.Data.SqlServerCe.SqlCeConnection MySqlCeConnection =
null;//数据库连接对象
        private System.Data.SqlServerCe.SqlCeCommand MySqlCeCommand = null;//
执行SQL指令的对象
        private System.Data.SqlServerCe.SqlCeTransaction MySqlCeTransaction =
null;//数据库的事务对象
        private System.Data.SqlServerCe.SqlCeDataReader MySqlCeDataReader =
null;
        private System.Data.SqlServerCe.SqlCeDataAdapter MySqlCeDataAdapter =
null;
        private System.Data.DataSet MySqlCeDataSet = null;
        private System.Data.DataView MySqlCeDataView = null;
        #endregion

        #region 构造函数
        //构造函数
        public SqlCeControl(string aDataBaseName, string aDataBaseConn)
        {
            //
```

```
            // TODO：在此处添加构造函数逻辑
            //
            //aDataBaseName：数据库文件名。
            //aDataBaseConn：数据库连接字符串。
            //如果只传文件名，则用默认的连接字符串，否则用用户传入的连接字符串。
            this.DataBaseName = aDataBaseName;
            if (aDataBaseConn == "")
            {
                this.DataBaseConn = "Data Source =" + this.DataBaseName;
            }
            else
            {
                this.DataBaseConn = aDataBaseConn;
            }//if
        }//SqlCeControl构造函数
        #endregion

        #region 释放资源
        //释放资源
        public void SqlCe_Dispose(bool disposing)
        {
            if (disposing)
            {
                if (this.MySqlCeEngine != null)
                {
                    this.MySqlCeEngine.Dispose();
                    this.MySqlCeEngine = null;
                }//MySqlCeEngine

                if (this.MySqlCeDataReader != null)
                {
                    this.MySqlCeDataReader.Dispose();
                    this.MySqlCeDataReader = null;
                }//MySqlCeDataReader

                if (this.MySqlCeTransaction != null)
                {
                    this.MySqlCeTransaction.Dispose();
                    this.MySqlCeTransaction = null;
                }//MySqlCeTransaction

                if (this.MySqlCeCommand != null)
                {
                    this.MySqlCeCommand.Connection.Close();
                    this.MySqlCeCommand.Dispose();
                    this.MySqlCeCommand = null;
                }//MySqlCeCommand

                if (this.MySqlCeConnection != null)
                {
                    this.MySqlCeConnection.Close();
```

```
                this.MySqlCeConnection.Dispose();
                this.MySqlCeConnection = null;
            }//MySqlCeConnection

            if (this.MySqlCeDataAdapter != null)
            {
                this.MySqlCeDataAdapter.Dispose();
                this.MySqlCeDataAdapter = null;
            }//MySqlCeDataAdapter

            if (this.MySqlCeDataSet != null)
            {
                this.MySqlCeDataSet.Clear();
                this.MySqlCeDataSet = null;
            }//MySqlCeDataSet

            if (this.MySqlCeDataView != null)
            {
                this.MySqlCeDataView.Dispose();
                this.MySqlCeDataView = null;
            }//MySqlCeDataView

        }//是否释放资源
    }//Dispose释放资源
    #endregion

    #region 建立数据库
    virtual public int SqlCe_CreateDataBase()
    {
        //使用try...catch机制防止系统出错
        try
        {
            //判断文件是否存在，如果在，则返回OP_DATABASEEXISTS
            if (File.Exists(this.DataBaseName))
            {
                return OP_DATABASEEXISTS;//用1表示执行不成功，已经存在该数据库
            }//
            else
            {
                this.MySqlCeEngine = new SqlCeEngine(this.DataBaseConn);//
创建MySqlCeEngine
                this.MySqlCeEngine.CreateDatabase();//建立数据库
                this.MySqlCeEngine.Dispose();//释放资源
                this.MySqlCeEngine = null;
                return OP_SUCCESS; //用0表示函数执行成功，建立了目标数据库
            }
        }
        catch
        {
            return OP_SYSTEMERR;//用返回值-1表示函数执行不成功，系统异常
        }
    }//SqlCe_CreateDataBase
```

```
    #endregion

    #region 删除数据库
    virtual public int SqlCe_DeleteDataBase()
    {
       //使用try...catch机制防止系统出错
       try
       {
          if (File.Exists(this.DataBaseName))
          {
             File.Delete(this.DataBaseName);
             return OP_SUCCESS;//用返回值0表示已经成功删除了该数据库
          }
          else
          {
             return OP_DATABASENOEXISTS;//用返回值2表示不存在该数据库文件
          }//if
       }
       catch
       {
          return OP_SYSTEMERR;//用返回值-1表示函数执行不成功，系统异常
       }//try...catch...
    }//SqlCe_DeleteDataBase
    #endregion
    #region 连接数据库
    private SqlCeConnection SqlCe_Connect()
  {
     try
     {
        SqlCeConnection tempSqlCeConnection=null;
        if(File.Exists(this.DataBaseName))
        {
           /
           if(this.MySqlCeConnection!=null)
           {
               this.SqlCe_Dispose(true);
           }
           tempSqlCeConnection = new
SqlCeConnection(this.DataBaseConn);
           return tempSqlCeConnection;
        }
        else
        {
           return null;
        }//if
     }
     catch
     {
        return null;
     }//try...catch...
  }//SqlCe_Connect
```

```
        #endregion

        #region 执行一组无返回的SQL语句
        virtual public int SqlCe_ExecuteNonReturnSql(string[] SqlCe_Sql)
        {
            //SQL指令组成组提交，有一个不成功就回滚
            try
            {
                this.MySqlCeConnection = this.SqlCe_Connect();//建立数据库连接
                if (this.MySqlCeConnection != null)
                {
                    try
                    {
                        this.MySqlCeConnection.Open();//打开连接
                        this.MySqlCeTransaction =
this.MySqlCeConnection.BeginTransaction();//建立数据库事务
                        this.MySqlCeCommand =
this.MySqlCeConnection.CreateCommand();//建立MySqlCeCommand对象
                        this.MySqlCeCommand.Transaction =
this.MySqlCeTransaction;
                        for (int i = 0; i < SqlCe_Sql.GetLength(0); i++)
                        {
                            this.MySqlCeCommand.CommandText = SqlCe_Sql[i];//传
入要执行的SQL语句
                            this.MySqlCeCommand.ExecuteNonQuery();//执行语句
                        }
                        this.MySqlCeTransaction.Commit();//提交事务
                        return OP_SUCCESS; //用返回值0表示事务提交成功
                    }
                    catch
                    {
                        this.MySqlCeTransaction.Rollback();//回滚事务
                        return OP_ROLLBACK; //用返回值3表示事务回滚成功
                    }//try...catch
                }
                else
                {
                    return OP_DATABASECONNECTIONERR;//用返回值-10表示数据库连接失
败
                }//if
            }
            catch
            {
                return OP_SYSTEMERR;//用返回值-1表示系统出错，系统异常
            }//try...catch
            finally
            {
                //事务已经提交，可以释放资源
                this.SqlCe_Dispose(true);
            }//try...catch...finally
        }//SqlCe_ExecuteNonReturnSql
        #endregion
```

```
        #region 返回一个SqlCeDataReader
        virtual public SqlCeDataReader
SqlCe_ExecuteReturnDataReaderSql(string SqlCe_SelectSql)
        {
            //注意：SqlCeDataReader是一个在线数据源，必须保持数据库连接才能操作该对象
            //如果返回NULL，可能连接失败，可能SQL语句出错
            try
            {
                this.MySqlCeConnection = this.SqlCe_Connect();//建立数据库连接
                if (this.MySqlCeConnection != null)
                {
                    this.MySqlCeConnection.Open();//打开连接
                    this.MySqlCeCommand =
this.MySqlCeConnection.CreateCommand();//建立MySqlCeCommand对象
                    this.MySqlCeCommand.CommandText = SqlCe_SelectSql;//传入要执
行的SQL语句
                    return this.MySqlCeCommand.ExecuteReader();
                }
                else
                {
                    return null;//如果返回NULL，可能连接失败
                }
            }
            catch
            {
                return null;//如果返回NULL，可能SQL语句出错
            }
            finally
            {
                //注意：这里不能释放连接及资源，必须等所有操作完成之后再由用户程序调用
SqlCe_Dispose(true)来释放资源
                this.SqlCe_Dispose(false);
            }
        }//SqlCe_ExecuteReturnDataReaderSql
        #endregion

        #region 返回一个SqlCeDataAdapter
        virtual public SqlCeDataAdapter
SqlCe_ExecuteReturnDataAdapterSql(string SqlCe_SelectSql)
        {
            //注意：SqlCeDataAdapter要求在线操作，即数据库连接不能断开，
            //如果返回NULL，可能连接失败，可能SQL语句出错
            try
            {
                SqlCeDataAdapter tempSqlCeDataAdapter = null;
                this.MySqlCeConnection = this.SqlCe_Connect();//建立数据库连接
                if (this.MySqlCeConnection != null)
```

```
                {
                    this.MySqlCeConnection.Open();//打开连接
                    tempSqlCeDataAdapter = new SqlCeDataAdapter(SqlCe_SelectSql,
this.MySqlCeConnection);
                    return tempSqlCeDataAdapter;
                }
                else
                {
                    return null;            //如果返回NULL，可能连接失败
                }
            }
            catch
            {
                return null;//如果返回NULL，可能SQL语句出错
            }
            finally
            {
                //注意：这里不能释放连接及资源，必须等所有操作完成之后再由用户程序调用
SqlCe_Dispose(true)来释放资源
                this.SqlCe_Dispose(false);
            }
        }//SqlCe_ExecuteReturnDataSetSql
        #endregion

        #region 返回一个DataSet
        virtual public DataSet SqlCe_ExecuteReturnDataSetSql(string
SqlCe_SelectSql)
        {
            //如果返回NULL，可能连接失败，可能SQL语句出错
            try
            {
                //这里可能有问题，tempSqlCeDataAdapter的资源如何释放？
                //SqlCeDataAdapter tempSqlCeDataAdapter=null;
                DataSet tempSqlCeDataSet = new DataSet();
                this.MySqlCeConnection = this.SqlCe_Connect(); //建立数据库连接
                if (this.MySqlCeConnection != null)
                {
                    this.MySqlCeConnection.Open();//打开连接
                    //tempSqlCeDataAdapter=new
SqlCeDataAdapter(sql,this.MySqlCeConnection);

//tempSqlCeDataAdapter.Fill(tempSqlCeDataSet,"MSqlCeDataSet");
                    this.MySqlCeDataAdapter = new
SqlCeDataAdapter(SqlCe_SelectSql, this.MySqlCeConnection);
                    this.MySqlCeDataAdapter.Fill(tempSqlCeDataSet,
"MySqlCeDataSet");
                    return tempSqlCeDataSet;
```

```
                }
                else
                {
                    return null;//如果返回NULL，可能连接失败
                }
            }
            catch
            {
                return null;//如果返回NULL，可能SQL语句出错
            }
            finally
            {
                //注意：这里不能释放连接及资源，必须等所有操作完成之后再由用户程序调用
SqlCe_Dispose(true)来释放资源
                this.SqlCe_Dispose(false);
            }
        }//SqlCe_ExecuteReturnDataSetSql
        #endregion

        #region 返回一个DataView
        virtual public DataView SqlCe_ExecuteReturnDataViewSql(string
SqlCe_SelectSql)
        {
            //如果返回NULL，可能连接失败，可能SQL语句出错
            try
            {
                //SqlCeDataAdapter tempSqlCeDataAdapter=null;
                //DataSet tempSqlCeDataSet=new DataSet();
                this.MySqlCeDataSet = new DataSet();
                DataView tempSqlCeDataView = null;
                this.MySqlCeConnection = this.SqlCe_Connect();//建立数据库连接
                if (this.MySqlCeConnection != null)
                {
                    this.MySqlCeConnection.Open();//打开连接
                    //tempSqlCeDataAdapter=new
SqlCeDataAdapter(SqlCe_SelectSql,this.MySqlCeConnection);

//tempSqlCeDataAdapter.Fill(tempSqlCeDataSet,"MySqlCeDataSet");
                    this.MySqlCeDataAdapter = new
SqlCeDataAdapter(SqlCe_SelectSql, this.MySqlCeConnection);
                    this.MySqlCeDataAdapter.Fill(this.MySqlCeDataSet,
"MySqlCeDataSet");
                    tempSqlCeDataView = new
DataView(this.MySqlCeDataSet.Tables["MySqlCeDataSet"]);
                    return tempSqlCeDataView;
                }
                else
                {
```

```
                return null;//如果返回NULL，可能连接失败
            }
        }
        catch
        {
            return null;//如果返回NULL，可能SQL语句出错
        }
        finally
        {
            //注意：这里不能释放连接及资源，必须等所有操作完成之后再由用户程序调用
SqlCe_Dispose(true)来释放资源
            this.SqlCe_Dispose(false);
        }
    }//SqlCe_ExecuteReturnDataViewSql
    #endregion
}// class SqlCeControl
#endregion

}//namespace CotrolSqlCe
```

12.3　操作 SqlServerCE 数据库实例

根据向导建立智能设备应用程序，在窗体中添加如下控件，如图 12-6 所示。

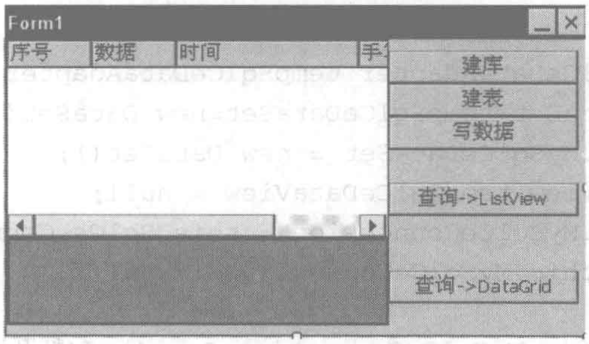


图 12-6

将上文中的 SqlCeControl.cs 文件加入本项目当中，Form1.cs 核心代码如下所示。

```
    SqlCeControl MyCE=new SqlCeControl("\\a.sdf");

private void button1_Click(object sender, System.EventArgs e)
{
    MyCE.SqlCe_CreateDataBase();
    MessageBox.Show("建库成功！");
}

private void button2_Click(object sender, System.EventArgs e)
{
```

```
        string [] sCreateTable=new string[3];
        //区域编码表
        sCreateTable[0]="Create table AreaTable(parentid nchar(20) primary
key,parentname nchar(100),childid nchar(20),childname nchar(100));";
        //表号基本信息
        sCreateTable[1]="Create table GasInfo(GasNo float primary key,GasMem
nchar(100));";
        //每月每表数据
        sCreateTable[2]="Create table GasMonth(GasNo float,Gasdata float, dt
datetime,HandWrite bit,IsUpdate bit,pid int IDENTITY (0, 1) primary key);";

        this.MyCE.SqlCe_ExecuteNonReturnSql(sCreateTable);
        MessageBox.Show("建表成功！");
    }

    private void button3_Click(object sender, System.EventArgs e)
    {
        System.DateTime d1=System.DateTime.Now;
        for(int i=0;i<100;i++)
        {
            string [] sInsert=new string[1];
            float fGasNo=123f;
            float Gasdata=123.4f;
            sInsert[0]="Insert into
GasMonth(GasNo,Gasdata,dt,HandWrite,IsUpdate)
values("+fGasNo+","+Gasdata+",'"+System.DateTime.Now+"',0,0);";

            this.MyCE.SqlCe_ExecuteNonReturnSql(sInsert);
        }
        System.DateTime d2=System.DateTime.Now;

    }




    private void button4_Click_1(object sender, System.EventArgs e)
    {

  this.dataGrid1.DataSource=this.MyCE.SqlCe_ExecuteReturnDataViewSql("select
count(*) from GasMonth");
    }

    private void button6_Click(object sender, System.EventArgs e)
    {
        System.DateTime d1=System.DateTime.Now;
        this.listView1.Items.Clear();
        System.Data.SqlServerCe.SqlCeDataReader aReader =
```

```
this.MyCE.SqlCe_ExecuteReturnDataReaderSql("select * from GasMonth where
pid=100");
            if(aReader!=null)
            {
                while(aReader.Read())
                {
                    this.listView1.Items.Add(new
ListViewItem(aReader.GetFloat(0).ToString()));
                    this.listView1.Items[this.listView1.Items.Count
-1].SubItems.Add(aReader.GetFloat(1).ToString());
                    this.listView1.Items[this.listView1.Items.Count
-1].SubItems.Add(aReader.GetDateTime(2).ToString());
                    this.listView1.Items[this.listView1.Items.Count
-1].SubItems.Add(aReader.GetBoolean(3).ToString());
                    this.listView1.Items[this.listView1.Items.Count
-1].SubItems.Add(aReader.GetBoolean(4).ToString());
                }
                aReader.Close();
            }
            this.MyCE.SqlCe_Dispose(true);
            System.DateTime d2=System.DateTime.Now;


        }
```

第 13 章　在 Windows CE 上进行打印

13.1　PCL 打印驱动的介绍

实际上，Windows CE 2.0 及其以上版本已经包括对打印机的支持。Windows CE 操作系统使用的打印模型是为基于 Windows CE 的桌面平台定义的打印模型的子集。只有少量为打印机驱动程序定义的图形驱动程序函数是 Windows CE 的打印机驱动程序所需要的。

Windows CE 的图形显示接口（GDI）和显示驱动程序完成与打印有关的大部分工作。在打印处理的开始，GDI 生成带属性的设备背景 deviec context，这些属性是在调用 DrvEnablePDEV 函数时从打印机驱动程序得到的。显示驱动程序用于提交（render）随后的绘图指令，该指令是由应用程序发向 DC 的：打印机驱动程序并不提交（render）这个文档。然后，GDI 把结果点阵图发送给打印机驱动程序，这样打印机驱动程序才能格式化点阵图并把它发送给打印机。为了节省内存，GDI 以批处理方式提交文档，也就是页面的水平不分：然后 GDI 多次调用打印机驱动程序把提交的文档发送给打印机。

基于 Windows CE 的打印机驱动程序只能实现一些图形驱动程序函数，这些函数是在搜集打印机度量标准、设置打印机、开始和结束打印任务和准备提交的打印内容时所必需的。中间过程中，打印机驱动程序还把点阵图数据从 GDI bitmap 格式转换为打印机所要求的格式。其中的操作包括把颜色简化到打印机的颜色空间、数据压缩、把数据转换成有时被称作打印机描述语言的打印机使用格式等。最后，打印机驱动程序调用监视器把提交的打印内容发送给打印机。

基于 Windows CE 的打印机驱动程序被编译为（.DLL）因此它必须输出 DrvEnableDriver 函数。

下面是打印机驱动程序必须实现的函数。

DrvCopyBits 由 GDI 调用用于把已提交的批次复制到打印机驱动程序。在调用函数 DrvDisablePDEV 时，打印机驱动程序指定在调用这个函数时要用到的点阵图格式。

DrvDisbalePDEV 用于在打印机设备环境不再需要时释放驱动程序使用的内存和资源。

DrvDisableSurfacd 由 GDI 调用用来通知打印机驱动程序为当前打印设备环境而生成的外表（Windws CE）不再支持。

DrvEnableDriver 从 GDI 接受两个回调函数的指针，并且为打印机驱动程序实现的其他的入口点返回 GDI 函数指针。

DrvEnablePdev 由 GDI 使用，用来为目标打印机收集打印机度量标准。打印机驱动程序返回在 GDINFO 结构中指定的设备。

DrvEndDoc 由 GDI 调用来结束或中途停止一个打印工作。

DrvGetModes 把打印机驱动程序支持的默认打印模式信息返回给 GDI。打印机驱动程序返回在 DEVMODE 结构中的默认配置。

DrvStartDoc 由 GDI 调用，来开始一个打印工作。

DrvStartPage 由 GDI 调用来开始一个打印工作的下一页。

Windows CE 使用注册键来存储设备的全局打印机设置和单个打印机驱动程序的配置信息。键值存储在注册表的 KEY_LOCAL_MACHINE\Printers\下。

存储在注册表中的全局打印机设置列举了打印使用的输入输入（I/O）口，提供了全局超时值以及存储了所选择的默认状态下的打印机驱动程序。默认打印机驱动程序存储为 DefaultPrinter 的值。打印机的接口存储为\Ports 子键的值。下面的注册文件摘录定义了 5 个打印机口，两个超时和默认的打印机驱动程序。

```
(HKEY_LOCAL_MACHINE\Printers\Ports)
"Port1"="COM1:9600"
"Port2"="COM1:57600"
"Port3"="IRDA"
"Port4"="LPT:"
"Port5"="NET0:"
(HKEY_LOCAL_MACHINE\Printers\Settings)
"TimeOut"=dword:1E
(HKEY_LOCAL_MACHINE\PrinterSettings)
"TimeOut"=dword:2D
(HKEY_LOCAL_MACHINE\Printers)
DefaultPrinter"="PCL Laster"
```

值分别为 9600 波特和 57600 波特的串口、一个红外口、一个并行口和一个网络打印机接口。对两个 COM1:口，分别指定了相应的波特。接口监视器分析了要提取的波特值和设备文件名。在 Windows CE2.0 中，COM3:口设定为红外。然而在 2.10 和以后的版本中心文件名“IRDA”预留为红外口，而 COM3:用于额外的串行口。在为打印分配红外口时，接口的键值必须是字符串“IRDA”，尽管该值并不是指定设备文件名的标准格式。名称“LPT1:”和“NET0:”分别预留为并行口和网络打印口，尽管这些名称中设备文件名索引可以是任何合法值。Windows CE 2.02 及其以后版本都使用了 TimeOut 键。该值是用秒测量的 16 进制数；例子是一个 30 s 的超时。在超时期间后，打印机驱动程序可以显示一个对话框让用户选择重试或取消。在这个例子中，默认打印机驱动程序“PCL Laser”是 KEY_LOCAL_MACHINE\Priners\中一个子键的名称。在 KEY_LOCAL_MACHINE\Priners\中 Windows CE 可以找到打印机控制语言（PCL）激光打印机驱动程序的配置。

对不使用并行口的打印机来说，如红外打印机或串行口打印机，让 Windows CE 决定打印机是否接受所有的打印数据是非常困难的。因此，超时参数用来在打印机不能发出任何其仍在活动的信号时侦测和报告打印错误。超时值存储在 PrintSetting\TimeOut 键中。默认值是 45 s。在 DefaultPrinter 键中存储的值是应用程序在通用打印对话框中显示给用户，让用户预选的打印机。

单个打印机驱动程序的设置存储在 KEY_LOCAL_MACHINE\Printers 键的子键中。打印机驱动程序的注册键应该定义为 4 个键：Driver、High Quality、Draft Quality 和 Color。其中 Driver 值列举过程包含打印机驱动程序的 DLL。如果打印机支持的话，Hight Quality 和 Draft Quality 的值存储高质量和绘图质量模式的分辨率。但并不是所有的打印机都支持绘图质量模式，因此 Draft Quality 键可以忽略。High Quality 键必须提供，它对应于打印机的最高质量模式。Color 值定义打印机是彩色的还是单色的。如果支持彩色的话，该键值必须设置为字符

"Color"。

下面是 PCL 激光打印机和 PCL 喷墨打印机驱动程序的设置例子：

```
(HEKY_LOCAL_MACHINE\Peinters\PCILaser)
"Driver"="pc1.Dll""High
Quality"="300"
"Draft Quality"="150"
"Color"=Monochrome"
"Version"="0x200"
(HKEY_LOCAL_MACHINE\Peinters\PCl Inkjet)
"Driver"="pcl.dll""High
Quality"="300""Draft
Quality"="150""Color"="Monochrome"
```

另外 Version 键可以用于支持打印机驱动程序服务多个打印机。

不同于 WINCE 以前的版本，WINCE6 的可视化组件中不再包含 PCL 打印机支持。但实际上 WINCE6 中自带对所有 PCL 打印机的驱动，只要符合 PCL 打印规范，无论是串口、并口、USB、红外、蓝牙，还是网络等打印方式，都可以被支持。首先确保通信接口驱动正确，然后添加环境变量"SYSGEN_PCL"即可驱动 PCL 打印机。

13.2　EVC 打印程序实例

根据 EVC 应用程序向导，建立基于对话框的应用程序工程，在对话框窗体上加入 3 个按钮，如图 13-1 所示。

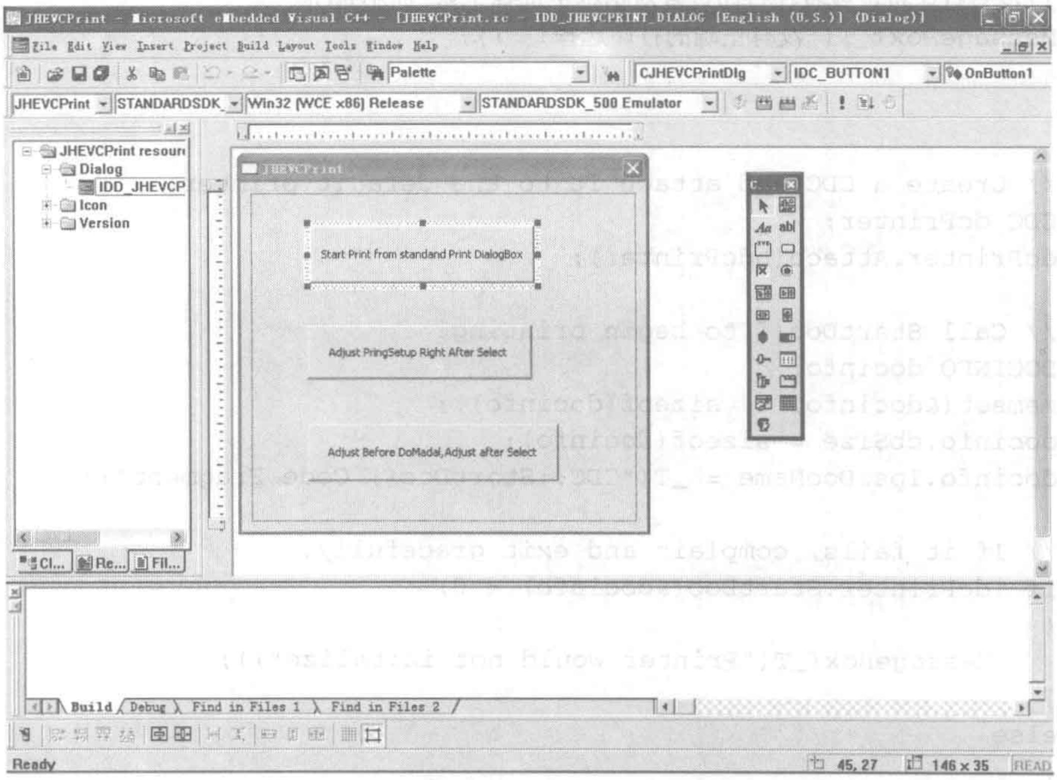


图 13-1

3 个按钮对应的代码如下所示：

```
void CJHEVCPrintDlg::OnButton1()
{

 //特别注意，不要草稿模式，草稿模式会很大的
 // TODO: Add your control notification handler code here
 ////////////////////////////////////////////////////////////////
 //以下两行OK
 // Get the default printer.
 //CPrintDialog dlg(FALSE);
 //dlg.GetDefaults();//得到默认的打印机

 //以下两行OK
 //以下两句弹出Windows的标准打印机设置对话框，用户可以自己选择
 CPrintDialog dlg(true);//让用户选择正确的打印机类型和端口号
 dlg.DoModal();
 //用户不可以选择
 //dlg.m_nFlags&=~PD_SELECTDRAFTMODE;

 // Is a default printer set up?
 HDC hdcPrinter = dlg.GetPrinterDC();


 if (hdcPrinter == NULL)
 {
     //没有打印机，或者在打印设置对话框中选择了×，而不是OK
     MessageBox(_T("没有正确的打印选择！"));
 }
 else
 {
     // Create a CDC and attach it to the default printer.
     CDC dcPrinter;
     dcPrinter.Attach(hdcPrinter);

     // Call StartDoc() to begin printing.
     DOCINFO docinfo;
     memset(&docinfo, 0, sizeof(docinfo));
     docinfo.cbSize = sizeof(docinfo);
     docinfo.lpszDocName = _T("CDC::StartDoc() Code Fragment");

     // If it fails, complain and exit gracefully.
     if (dcPrinter.StartDoc(&docinfo) < 0)
     {
         MessageBox(_T("Printer would not initalize"));
     }
     else
     {
     // Start a page.
      if (dcPrinter.StartPage() < 0)
       {
```

```
        MessageBox(_T("Could not start page"));
        dcPrinter.AbortDoc();
    }
    else
    {
        // Do some printing.
        //黑白打印，大字

        CFont newFont;
        newFont.CreateFont(
            30,                     // nHeight
            0,                      // nWidth
            0,                      // nEscapement
            0,                      // nOrientation
            FW_NORMAL,              // nWeight
            FALSE,                  // bItalic
            FALSE,                  // bUnderline
            0,                      // cStrikeOut
            ANSI_CHARSET,           // nCharSet
            OUT_DEFAULT_PRECIS,     // nOutPrecision
            CLIP_DEFAULT_PRECIS,    // nClipPrecision
            DEFAULT_QUALITY,        // nQuality
            DEFAULT_PITCH | FF_SWISS,  // nPitchAndFamily
            _T("Arial"));

        CFont* pOldFont = dcPrinter.SelectObject(&newFont);
        // CGdiObject* pOldFont = dcPrinter.SelectStockObject(SYSTEM_FONT);

        dcPrinter.ExtTextOut(150, 150,ETO_CLIPPED,NULL,_T("蒋辉不是个大傻
瓜!"),8,NULL);
        dcPrinter.ExtTextOut(50, 50,ETO_CLIPPED,NULL,_T("Hello World!"),
12,NULL);
        dcPrinter.SelectObject(pOldFont);
        newFont.DeleteObject();


        //大字体，彩色
          CFont newFont1;
        newFont1.CreateFont(
            100,                    // nHeight
            0,                      // nWidth
            0,                      // nEscapement
            0,                      // nOrientation
            FW_NORMAL,              // nWeight
            FALSE,                  // bItalic
            FALSE,                  // bUnderline
            0,                      // cStrikeOut
            ANSI_CHARSET,           // nCharSet
            OUT_DEFAULT_PRECIS,     // nOutPrecision
            CLIP_DEFAULT_PRECIS,    // nClipPrecision
            DEFAULT_QUALITY,        // nQuality
```

```
                DEFAULT_PITCH | FF_SWISS, // nPitchAndFamily
                _T("Arial"));

          pOldFont = dcPrinter.SelectObject(&newFont1);
          //标准字体,彩色
          dcPrinter.SetTextColor(RGB(0,255,0));
          dcPrinter.ExtTextOut(400, 550,ETO_CLIPPED,NULL,_T("蒋辉不是个大傻
瓜!"),8,NULL);
          dcPrinter.SelectObject(pOldFont);
          newFont1.DeleteObject();


          //黑色线
          dcPrinter.MoveTo(200,200);
          dcPrinter.LineTo(400,400);
          //彩色线，变宽
          CPen newPen(PS_SOLID,10,RGB(0,255,0));
          CPen* pOldPen=dcPrinter.SelectObject(&newPen);
          dcPrinter.MoveTo(750,750);
          dcPrinter.LineTo(950,950);
          dcPrinter.SelectObject(pOldPen);
          newPen.DeleteObject();

          //彩色打印矩形
          CBrush brush2(RGB(255,0,0));
          CRect rc(300,300,400,400);
          dcPrinter.FillRect(&rc,&brush2);
          brush2.DeleteObject();


          dcPrinter.EndPage();
          dcPrinter.EndDoc();

       }
    }
    CDC::FromHandle(hdcPrinter)->DeleteDC();//N非ULL才能释放
  }
  //CDC::FromHandle(hdcPrinter)->DeleteDC();//N非ULL才能释放
  //////////////////////////////////////////////////////

 }


 void jhMessageInt(int input)
 {
  char buffer[100];
  sprintf(buffer,"%d",input);
  CString s=buffer;
  AfxMessageBox(s,NULL,MB_OK);
 }
```

```
bool CJHEVCPrintDlg::AdjustRightAfterSelect(CString DeviceName,CString
PortName,DWORD dwFlags)
{
  //打印机
  if (DeviceName!="PCL Inkjet"){
      MessageBox(_T("打印机必须选择PCL_Inkjet"));
      return false;
  }
  //端口
  if (PortName!="LPT1:"){
      MessageBox(_T("端口必须选择LPT1:,如果下拉列表没有LPT1:,请检查打印机是否连线正
确并上电"));
      return false;
  }
  //纸张大小
  if ((dwFlags & PD_SELECTA4)==0){
      MessageBox(_T("必须选择A4幅面纸张"));
      return false;
  }
  //草稿模式
  if ((dwFlags & PD_SELECTDRAFTMODE)!=0){
      MessageBox(_T("不能选择草稿模式"));
      return false;
  }
  //彩色
  if ((dwFlags & PD_SELECTPRINTINCOLOR)==0){
      MessageBox(_T("必须选择彩色模式"));
      return false;
  }
  //打印全部
  if ((dwFlags & PD_SELECTALLPAGES)==0){
      MessageBox(_T("必须选择全部打印"));
      return false;
  }
  //横向打印
  if ((dwFlags & PD_SELECTPORTRAIT)==0){
      MessageBox(_T("必须选择纵向打印"));
      return false;
  }


  return true;
}


void CJHEVCPrintDlg::OnButton2()
{
  //CPrintDialog dlg(true);//让用户选择正确的打印机类型和端口号
  //dlg.DoModal();

  //dlg.m_nMinPage=6; //test,OK
```

```
    //dlg.m_nMaxPage=27; //test,OK

    //MessageBox(dlg.GetDeviceName());//这个能够准确得到用户选择的下拉列表框中选择的
打印机型号名称
    //用户不同的选择，这里给出的实际是ListBox_DeviceName.Text,OK

    //MessageBox(dlg.GetPortName());//这个能够准确得到用户选择的下拉列表框中选择的打印
机端口名称
    //用户不同的选择，这里给出的实际是ListBox_PortName.Text,OK

    //CSize PaperSize=dlg.GetPaperSize();
    //这个能够准确得到用户选择的下拉列表框中选择的打印机纸张大小
    //用户不同的选择，这里给出的实际是纸张的cx和cy,
    //但是不论我们如何选择纸张下拉框的几种选择，cx总是=8500,cy总是=11000,OK


    /*
    DWORD dwFlags=dlg.m_pd.dwFlags;
    //是否选择草稿模式,关键参数参考
    //"C:\Program Files\Windows CE
Tools\wce500\STANDARDSDK_500\Include\x86\commdlg.h"
    if ((dwFlags & PD_SELECTDRAFTMODE)!=0){
        MessageBox(_T("Draft Mode"));
    }
    else{
            MessageBox(_T("not Draft Mode"));
    }
    *///这个得到了是否选择草稿模式，非常难得，自此，应该都可以解决，所以我们采用事后判断方式
定制打印机


    CPrintDialog dlg(true);//让用户选择正确的打印机类型和端口号
    if (dlg.DoModal()==IDOK){
        //必须选择"确定"
        CString DeviceName=dlg.GetDeviceName();
        CString PortName=dlg.GetPortName();
        DWORD dwFlags=dlg.m_pd.dwFlags;
        if (this->AdjustRightAfterSelect(DeviceName,PortName,dwFlags)){
            //必须正确判断才打印
            //MessageBox(_T("Setup Right"));
            /////////////////////////////////////////////////////////
            HDC hdcPrinter = dlg.GetPrinterDC();
            if (hdcPrinter == NULL)
            {
                //没有打印机，或者在打印设置对话框中选择了×，而不是OK
                MessageBox(_T("打印机配置不正确"));
            }
            else
            {
                // Create a CDC and attach it to the default printer.
                CDC dcPrinter;
                dcPrinter.Attach(hdcPrinter);
```

```
        // Call StartDoc() to begin printing.
        DOCINFO docinfo;
        memset(&docinfo, 0, sizeof(docinfo));
        docinfo.cbSize = sizeof(docinfo);
        docinfo.lpszDocName = _T("CDC::StartDoc() Code Fragment");

        // If it fails, complain and exit gracefully.
        if (dcPrinter.StartDoc(&docinfo) < 0)
        {
            MessageBox(_T("Printer would not initalize"));
        }
        else
        {
        // Start a page.
          if (dcPrinter.StartPage() < 0)
          {
            MessageBox(_T("Could not start page"));
            dcPrinter.AbortDoc();
          }
          else
          {
            // Do some printing.
            //黑白打印，大字

            CFont newFont;
            newFont.CreateFont(
                  30,                        // nHeight
                  0,                        // nWidth
                  0,                        // nEscapement
                  0,                        // nOrientation
                  FW_NORMAL,                 // nWeight
                  FALSE,                     // bItalic
                  FALSE,                     // bUnderline
                  0,                        // cStrikeOut
                  ANSI_CHARSET,               // nCharSet
                  OUT_DEFAULT_PRECIS,          // nOutPrecision
                  CLIP_DEFAULT_PRECIS,         // nClipPrecision
                  DEFAULT_QUALITY,            // nQuality
                  DEFAULT_PITCH | FF_SWISS,    // nPitchAndFamily
                  _T("Arial"));

            CFont* pOldFont = dcPrinter.SelectObject(&newFont);
           // CGdiObject* pOldFont =
dcPrinter.SelectStockObject(SYSTEM_FONT);

            dcPrinter.ExtTextOut(150, 150,ETO_CLIPPED,NULL,_T("蒋辉是个
大傻瓜！"),8,NULL);
            dcPrinter.ExtTextOut(50, 50,ETO_CLIPPED,NULL,_T("Hello
World!"), 12,NULL);
            dcPrinter.SelectObject(pOldFont);
            newFont.DeleteObject();
```

```
//大字体，彩色
CFont newFont1;
newFont1.CreateFont(
    100,                    // nHeight
    0,                      // nWidth
    0,                      // nEscapement
    0,                      // nOrientation
    FW_NORMAL,              // nWeight
    FALSE,                  // bItalic
    FALSE,                  // bUnderline
    0,                      // cStrikeOut
    ANSI_CHARSET,           // nCharSet
    OUT_DEFAULT_PRECIS,     // nOutPrecision
    CLIP_DEFAULT_PRECIS,    // nClipPrecision
    DEFAULT_QUALITY,        // nQuality
    DEFAULT_PITCH | FF_SWISS, // nPitchAndFamily
    _T("Arial"));

pOldFont = dcPrinter.SelectObject(&newFont1);
//标准字体,彩色
dcPrinter.SetTextColor(RGB(0,255,0));
dcPrinter.ExtTextOut(400, 550,ETO_CLIPPED,NULL,_T("蒋辉是个
大傻瓜!"),8,NULL);
dcPrinter.SelectObject(pOldFont);
newFont1.DeleteObject();

//黑色线
dcPrinter.MoveTo(200,200);
dcPrinter.LineTo(400,400);
//彩色线，变宽
CPen newPen(PS_SOLID,10,RGB(0,255,0));
CPen* pOldPen=dcPrinter.SelectObject(&newPen);
dcPrinter.MoveTo(750,750);
dcPrinter.LineTo(950,950);
dcPrinter.SelectObject(pOldPen);
newPen.DeleteObject();

//彩色打印矩形
CBrush brush2(RGB(255,0,0));
CRect rc(300,300,400,400);
dcPrinter.FillRect(&rc,&brush2);
brush2.DeleteObject();

dcPrinter.EndPage();
dcPrinter.EndDoc();
```

```
            }
        }
        CDC::FromHandle(hdcPrinter)->DeleteDC();//N非ULL才能释放
    }
    //CDC::FromHandle(hdcPrinter)->DeleteDC();//N非ULL才能释放
    //////////////////////////////////////////////////////

  }

}

}

void CJHEVCPrintDlg::OnButton3()
{
 // TODO: Add your control notification handler code here
 //我们首先调整到我们默认的值，有些对话框Disable
 CPrintDialog dlg(true);//让用户选择正确的打印机类型和端口号


 if (dlg.DoModal()==IDOK){
    //必须选择"确定"
    CString DeviceName=dlg.GetDeviceName();
    CString PortName=dlg.GetPortName();
    DWORD dwFlags=dlg.m_pd.dwFlags;
    if (this->AdjustRightAfterSelect(DeviceName,PortName,dwFlags)){
        //必须正确判断才打印
        //MessageBox(_T("Setup Right"));
        //////////////////////////////////////////////////////
        HDC hdcPrinter = dlg.GetPrinterDC();
        if (hdcPrinter == NULL)
        {
            //没有打印机，或者在打印设置对话框中选择了×，而不是OK
            MessageBox(_T("打印机配置不正确"));
        }
        else
        {
            // Create a CDC and attach it to the default printer.
            CDC dcPrinter;
            dcPrinter.Attach(hdcPrinter);

            // Call StartDoc() to begin printing.
            DOCINFO docinfo;
            memset(&docinfo, 0, sizeof(docinfo));
            docinfo.cbSize = sizeof(docinfo);
```

```
        docinfo.lpszDocName = _T("CDC::StartDoc() Code Fragment");

        // If it fails, complain and exit gracefully.
        if (dcPrinter.StartDoc(&docinfo) < 0)
        {
            MessageBox(_T("Printer would not initalize"));
        }
        else
        {
        // Start a page.
          if (dcPrinter.StartPage() < 0)
          {
             MessageBox(_T("Could not start page"));
             dcPrinter.AbortDoc();
          }
          else
          {
             // Do some printing.
             //黑白打印，大字
             CFont newFont;
             newFont.CreateFont(
                   30,                       // nHeight
                   0,                        // nWidth
                   0,                        // nEscapement
                   0,                        // nOrientation
                   FW_NORMAL,                // nWeight
                   FALSE,                    // bItalic
                   FALSE,                    // bUnderline
                   0,                        // cStrikeOut
                   ANSI_CHARSET,             // nCharSet
                   OUT_DEFAULT_PRECIS,       // nOutPrecision
                   CLIP_DEFAULT_PRECIS,      // nClipPrecision
                   DEFAULT_QUALITY,          // nQuality
                   DEFAULT_PITCH | FF_SWISS, // nPitchAndFamily
                   _T("Arial"));

             CFont* pOldFont = dcPrinter.SelectObject(&newFont);
            // CGdiObject* pOldFont =
dcPrinter.SelectStockObject(SYSTEM_FONT);

             dcPrinter.ExtTextOut(150, 150,ETO_CLIPPED,NULL,_T("蒋辉不是
个大傻瓜!"),8,NULL);
             dcPrinter.ExtTextOut(50, 50,ETO_CLIPPED,NULL,_T("Hello
World!"), 12,NULL);
             dcPrinter.SelectObject(pOldFont);
             newFont.DeleteObject();
```

```
//大字体，彩色
CFont newFont1;
newFont1.CreateFont(
    100,                          // nHeight
    0,                           // nWidth
    0,                           // nEscapement
    0,                           // nOrientation
    FW_NORMAL,                    // nWeight
    FALSE,                       // bItalic
    FALSE,                       // bUnderline
    0,                           // cStrikeOut
    ANSI_CHARSET,                 // nCharSet
    OUT_DEFAULT_PRECIS,            // nOutPrecision
    CLIP_DEFAULT_PRECIS,           // nClipPrecision
    DEFAULT_QUALITY,              // nQuality
    DEFAULT_PITCH | FF_SWISS,  // nPitchAndFamily
    _T("Arial"));

pOldFont = dcPrinter.SelectObject(&newFont1);
//标准字体,彩色
dcPrinter.SetTextColor(RGB(0,255,0));
dcPrinter.ExtTextOut(400, 550,ETO_CLIPPED,NULL,_T("蒋辉是个
大傻瓜!"),8,NULL);
dcPrinter.SelectObject(pOldFont);
newFont1.DeleteObject();


//黑色线
dcPrinter.MoveTo(200,200);
dcPrinter.LineTo(400,400);
//彩色线，变宽
CPen newPen(PS_SOLID,10,RGB(0,255,0));
CPen* pOldPen=dcPrinter.SelectObject(&newPen);
dcPrinter.MoveTo(750,750);
dcPrinter.LineTo(950,950);
dcPrinter.SelectObject(pOldPen);
newPen.DeleteObject();


//彩色打印矩形
CBrush brush2(RGB(255,0,0));
CRect rc(300,300,400,400);
dcPrinter.FillRect(&rc,&brush2);
brush2.DeleteObject();
```

```
                dcPrinter.EndPage();
                dcPrinter.EndDoc();

            }
        }
        CDC::FromHandle(hdcPrinter)->DeleteDC();//N非ULL才能释放
    }
    //CDC::FromHandle(hdcPrinter)->DeleteDC();//N非ULL才能释放
    ////////////////////////////////////////////////////////////

  }

}
```

Document generated by Anna's Archive around 2023-2024 as part of the DuXiu collection (https://annas-blog.org/duxiu-exclusive.html).

Images have been losslessly embedded. Information about the original file can be found in PDF attachments. Some stats (more in the PDF attachments):
{
"filename": "44CKV2luZG93cyBDRTblvIDlj5Hnu4/lhbjjgItfMTIyNDM5MTQuemlw",
"filename_decoded": "\u300aWindows CE6\u5f00\u53d1\u7ecf\u5178\u300b_12243914.zip",
"filesize": 81405569,
"md5": "6d5e2879149d5ac98b5ea6e06e71ce60",
"header_md5": "f82dfd8f4fa2ef6e3209eb8c95fc74e1",
"sha1": "f746bda1f9be3ecdc9a588f38f693757c1004b2c",
"sha256": "b7a3bf0641f0aa1d88b856ed00443d70edd2de7bf36e80debd0d1f478d0a01b4",
"crc32": 3320123365,
"zip_password": "",
"uncompressed_size": 87330447,
"pdg_dir_name": "\u00ed\u2562Windows CE6\u2510\u00ac\u2556\u00f3\u255b\u00a1\u2561\u03a3\u00ed\u2556_12243914",
"pdg_main_pages_found": 338,
"pdg_main_pages_max": 338,
"total_pages": 346,
"total_pixels": 2252639920,
"pdf_generation_missing_pages": false
}